УДК 004.75

Ю.А. Алдунин

ОРИЕНТИРОВАННЫЕ НА КЛИЕНТА МОДЕЛИ НЕПРОТИВОРЕЧИВОСТИ И СПОСОБЫ ИХ РЕАЛИЗАЦИИ

Рассматриваются основные модели непротиворечивости ориентированные на клиента. Приводятся способы их реализации и описываются основные проблемы реализации, а также пути их решения.

Введение. Существует большое количество моделей непротиворечивости, ориентированных на данные. Они позволяют поддерживать непротиворечивость хранилища данных в условиях, когда множество параллельных процессов вносит изменения в эти данные. Их краткая характеристика была дана в предыдущей статье («Синхронизация времени в распределенных системах в зависимости от выбранной модели непротиворечивости»). Рассмотрим теперь случай, когда одновременное изменение данных отсутствует, либо возникающие конфликты легко разрешимы. Модели непротиворечивости, относящиеся к таким базам данных, являются очень слабыми, но это не мешает им выполнять свои функции.

Цель работы – проанализировать и систематизировать существующие модели непротиворечивости, ориентированные на клиента, а также способы их реализации и основные проблемы, при этом возникающие.

Теоретическая часть. Остановимся подробнее на моделях непротиворечивости, ориентированных на клиента.

Потенциальная непротиворечивость

В большинстве случаев изменение данных осуществляется одним или несколькими процессами, остальные лишь просматривают либо анализируют эти данные. Основной задачей в данном случае является обеспечение быстрого доведения изменений до всех потребителей. Рассмотрим, например, DNS. Все пространство имен разделено на домены. Каждый домен имеет владельца, ответственного за него и вносящего изменения, таким образом, конфликт одновременной записи невозможен. Единственная проблема – возможный конфликт чтения-записи, когда данные уже изменены, но, поскольку изменения распространяются постепенно, эти изменения будут доступны всем пользователям только через некоторый промежуток времени.

Подобные системы характеризуются нечувствительностью к высокой степени нарушения непротиворечивости. Изменения в них относительно редки, и все копии постепенно становятся непротиворечивыми. Такая форма непротиворечивости называется потенциальной непротиворечивостью.

Фактически единственное требование для обеспечения потенциальной непротиворечивости – гарантированное распространение изменений по всем копиям базы данных. Конфликты множественной записи легко разрешаются, так как изменять данные может лишь ограниченный круг пользователей.

Хранилища, обладающие потенциальной непротиворечивостью, имеют один существенный недостаток: если пользователь вынужден работать с разными копиями базы данных, то весьма вероятна ситуация, когда в более поздний момент времени он, подключившись к другой реплике, может обнаружить более старые данные. Иными словами, пользователь работает с одной репликой, вносит изменения; затем, к примеру, переезжает в другой офис, снова подключается к базе данных и обнаруживает, что его изменения отсутствуют.

Этот пример типичен для баз данных с потенциальной непротиворечивостью. Для борьбы с подобными явлениями прибегают к помощи ориентированной на клиента непротиворечивости. Данные модели непротиворечивости дают конкретному пользователю гарантию непротиворечивости при доступе к базе данных. Относительно других пользователей никаких гарантий не предоставляется.

Непротиворечивость монотонного чтения

Первая из моделей непротиворечивости, ориентированных на клиента, – модель монотонного чтения. Непротиворечивость монотонного чтения обеспечивается, если база данных удовлетворяет условию, что если процесс читает значение элемента данных x, то любое последующее чтение x всегда возвращает то же либо

более позднее значение. То есть непротиворечивость монотонного чтения должна обеспечивать следующее: если в некоторый момент времени процесс видит некое значение x, то позднее он никогда не увидит более старого значения x.

Примером данного типа непротиворечивости может служить система электронной почты. Почтовый ящик пользователя является своего рода репликой распределенной базы данных, изменения в которую вносятся нечасто (по запросу). Получив электронную почту (обновив локальную реплику), пользователь видит содержимое своего почтового ящика. Позже, переместившись в другой офис, он вновь обновляет локальную реплику и видит те же самые сообщения, что он видел ранее, и (при наличии) более новые сообщения.

Если же возникает ситуация, когда клиент, подключившись к другой реплике, обнаруживает более старые данные (например, приехав в другой офис, он получает сообщения электронной почты, которые уже удалил), непротиворечивость монотонного чтения нарушается.

Непротиворечивость монотонной записи

В большинстве случаев важно, чтобы операции записи распространялись по всем копиям базы данных в правильной последовательности. Это обеспечивается хранилищами, обладающими свойством непротиворечивости монотонной записи.

Если обеспечивается условие, что любая операция записи в элемент данных x завершается раньше любой из последующих операций записи этого же клиента в элемент x, то база данных обладает свойством непротиворечивости монотонной записи. Иными словами, операция записи в элемент данных x осуществляется, только если эта копия соответствует результатам предыдущей операции записи, выполненной, возможно, на другой копии x.

Актуализация копии x не важна, если каждая последующая операция записи полностью замещает текущее значение x. Но операции записи очень часто выполняются только над частью элемента данных. Примером может служить библиотечный файл, состоящий из набора функций. В подавляющем большинстве случаев изменения такого файла заключаются в изменении одной или нескольких функций. Если все изменения вносились последовательно, как того требует непротиворечивость монотонной записи, то результатом будет работоспособная актуальная версия файла, содержащая в себе все предыдущие изменения.

Если же возникает ситуация, когда в любую реплику осуществляется операция записи до того как в нее же были осуществлены все предыдущие операции записи, инициированные данным клиентом, то непротиворечивость монотонной записи нарушается.

Чтение собственных записей

На непротиворечивость монотонной записи похожа еще одна модель непротиворечивости – чтение собственных записей. База данных обладает свойством непротиворечивости чтения собственных записей, если выполняется условие: результат записи, инициированной процессом в элемент данных x, всегда виден последующим операциям чтения элемента x этого же процесса. То есть любая операция записи завершается ранее любой следующей операции чтения того же процесса, на какой бы из копий базы данных они не выполнялись.

Для иллюстрации данной модели непротиворечивости можно привести пример редактирования web-страниц. После редактирования web-страница помещается обратно на сервер, после чего вновь просматривается с помощью web-браузера. Но предыдущий вариант страницы может оказаться в кэше web-браузера, и пользователь не увидит своих изменений. Если же выполняется требование непротиворечивости чтения собственных записей, то версия файла в кэше будет признана устаревшей и произойдет повторная загрузка страницы с web-сервера и пользователь увидит результат своего труда.

Запись за чтением

Еще одна модель ориентированной на клиента непротиворечивости – непротиворечивость записи за чтением. Хранилище данных обеспечивает непротиворечивость записи за чтением, если операция записи в элемент данных x, инициированная процессом, всегда выполняется над тем же самым или более актуальным значением x, которое было прочитано предыдущей операцией процесса. Или иначе: любая операция записи в элемент данных x, производимая процессом, будет выполняться над копией x, содержащей последнее считанное этим же процессом значение x.

Применение непротиворечивости записи за чтением позволяет гарантировать, что при использовании сетевых групп новостей пользователь увидит письма с ответами только после получения оригинального письма. То есть если пользователь сначала читает письмо А, а потом отвечает на него письмом Б, то письмо Б будет разослано всем получателям только после письма А (все пользователи гарантированно получат сначала письмо А, содержащее вопрос, и лишь затем письмо Б, содержащее ответ на него, в

противном случае могла бы возникнуть путаница).

Рассмотрим теперь способы реализации указанных моделей непротиворечивости.

Если не принимать в расчет вопросы производительности, то реализация непротиворечивости, ориентированной на клиента, достаточно проста.

Каждой операции чтения и записи присваивается уникальный глобальный идентификатор. Присвоение осуществляется тем сервером, который первым выполняет соответствующую операцию. Для каждого клиента существует два набора идентификаторов: чтения и записи.

Реализация непротиворечивости монотонного чтения заключается в следующем. При попытке чтения данных клиентом сервер проверяет его набор идентификаторов записи. Если в локальной копии данных присутствует результат всех его операций записи, то данные возвращаются клиенту. Если это не так, то инициируется актуализация требуемых данных. Возможен и второй вариант: запрос на чтение передается серверу, содержащему все актуальные результаты (на котором уже выполнены все операции записи, присутствующие в наборе клиента). Для повышения эффективности идентификаторы операций могут содержать данные сервера, выполнившего операцию записи первым, что значительно ускорит возможную актуализацию данных (так достоверный источник в данном случае известен заранее). Если элемент данных модифицируется только его владельцем, то последний может взять формирование идентификаторов на себя.

Аналогично реализуется и непротиворечивость монотонной записи. Сервер удостоверяется, что результаты всех предшествующих операций записи данного процесса присутствуют в его реплике, а также сохранен порядок выполнения операций (если это не так, то сервер либо актуализирует свою копию базы данных путем выполнения операций записи, либо перенаправляет клиента на другой сервер, уже содержащий все требуемые данные). После чего операция выполняется, и ей присваивается идентификатор.

Непротиворечивость чтения собственных записей требует, чтобы сервер, выполняющий операцию чтения, содержал все записанные клиентом данные. Если это не так, то сервер запрашивает все отсутствующие данные и обрабатывает запрос. Для оптимизации чтения возможно обращение к другой копии базы данных, уже содержащей все записи клиента.

Непротиворечивость записи за чтением реализуется следующим образом. Сервер запрашивает все отсутствующие данные, прочитанные клиентом (если они отсутствуют), и осуществляет запись.

Со временем наборы идентификаторов клиентов могут стать слишком большими. Чтобы этого не допустить, операции чтения и записи группируются в сеансы. Начало сеанса связывают с запуском приложения, а конец – с закрытием, после чего оперируют (для завершенных сеансов) идентификаторами сеансов, а не единичных операций.

Главная проблема реализации вытекает из способа представления наборов идентификаторов. Каждый набор состоит из множества идентификаторов операций. Этот набор передается на сервер каждый раз вместе с запросом на чтение или запись для того, чтобы убедиться в актуальности обрабатываемых данных. Для решения этой проблемы применяют механизм векторных отметок времени, хранящихся на сервере и содержащих информацию о временной отметке всех операций для конкретного инициатора (первичного исполнителя).

Вывод. Ориентированность на клиента снимает ряд ограничений с реализации распределенной базы данных. Позволяет добиться большего быстродействия, так как используются более слабые модели непротиворечивости, по сравнению моделями, ориентированными на данные. Также мы видим, что в зависимости от поставленных задач и имеющихся ограничений, существует возможность подобрать требуемую модель непротиворечивости, максимально учитывающую специфику конкретной задачи. Все приведенные модели успешно применяются на практике. А имеющиеся проблемы реализации успешно преодолеваются.

Библиографический список

1. Terry, D., Demers, A., Petersen, K., Spreitzer, M., Theimer, M., and Welsh, B.: “Session Guarantees for Weakly Consistent Replicated Data.” Proc. Trird Int’l Conf on Parallel and Distributed Information Systems. IEEE, 1994. pp. 140-149.

2. Terry, D., Petersen, K., Spreitzer, M., and Theimer, M.: “The Case for Non-transparent Replication: Examples from Bayou.” IEEE Data Engineering, vol. 21, no. 4, pp. 12-20, Dec. 1998.

3. Birrell, A., Levin, R., Needham, R., and Schroeder, M.: “Grapevine: An Exercise in Distributed Computing.” Commun. ACM, vol. 25, no. 4, pp. 260-274, Apr. 1982.