Проектирование распределённых систем обработки объектных структур данных♣

© А.А. Демидов

Исследовательский центр искусственного интеллекта ИПС РАН
alex@dem.botik.ru

Аннотация

Статья посвящена проблеме обеспечения устойчивости к отказам и согласованности изменений сложно структурированных данных при конкурирующем доступе к ним транзакций распределённой системы. Некоторые распространённые принципы проектирования распределённых хранилищ данных формализуются заново с целью анализа необходимости их применения в тех или иных случаях, обусловленных требованиями предметной области реальных задач. Особое внимание уделено проблеме выбора оптимальной стратегии обеспечения непротиворечивости данных.

1 Введение

Распределённые системы обработки и хранения данных позволяют оптимальным образом решать задачи объединения территориально разобщённых узлов в единый информационно-вычислительный комплекс. Технологии, лежащие в основе таких решений, как правило, довольно сложны. Причиной тому являются физические ограничения среды функционирования распределённой системы: конечная надёжность и пропускная способность как самих узлов, так и каналов передачи данных, что делает необходимым применение алгоритмов коррекции аппаратных сбоев, репликации и восстановления данных, синхронизации процессов доступа к ресурсам, балансировки нагрузки и организации управления системой [1, 7].

Рассмотрим элементарную распределённую систему: сеть пронумерованных узлов, хранящих простые байты данных; пусть каждый байт в такой системе уникально адресуется номером узла и смещением элемента данных в его в памяти. Если надёжность и пропускная способность компонентов такой системы не ограничены, то никаких дополнительных алгоритмов здесь не требуется – можно свободно читать и писать отдельные байты, состояние такой простой системы останется непротиворечивым, поскольку байты независимы друг от друга, а каждый байт записывается **атомарно (Atomic)**, то есть неделимым на отдельные шаги образом.

Теперь усложним систему – пусть элементами информации в ней будут непересекающиеся последовательности байт. При конкурирующем доступе здесь возможны конфликты совместного доступа к данным. Они вызваны потерей атомарности операций с **объектом** – элементом данных сложной структуры, атомарность операций с которым не обеспечивается средствами системы нижнего уровня (например, операционной системы или СУБД). Если один процесс начал запись объекта в тот момент, когда другой процесс не закончил чтение данного объекта, то происходит конфликт **«чтение – запись»** – второй процесс получит **противоречивую (Non-Consistent)** копию объекта, реально содержащую части разных объектов – до записи и после. Так же, если два процесса начнут запись одного объекта в одно и то же время, то возникает конфликт **«запись – запись»** – информация в системе в таком случае будет необратимо испорчена.

Между тем, реальные объекты в настоящей системе редко бывают независимы друг от друга – как правило, они имеют связи или же логические взаимозависимости своих данных. Поэтому модель усложняется ещё более – объектами в ней являются уже пересекающиеся последовательности байт. Область записи и область чтения объекта в такой модели уже не всегда совпадают, область чтения одного объекта может являться областью записи другого, объекты могут пересекаться только по чтению или только по записи – всё это допустимые варианты организации данных. Конфликты **«запись – запись»** в таком случае можно подразделить на физические, или **низкоуровневые**, связанные с разрушением данных в общей области записи объектов, и логические, **высокоуровневые**, вызванные несогласованной модификацией данных в области чтения другого объекта [3].

При произвольном изменении объекта пересекающиеся с ним объекты пострадают, однако если пишущему процессу известно состояние всех пересекающихся объектов, то противоречивости при записи можно избежать, если изменить объект согласованно на основе совокупной информации. Это возможно только в случае обеспечения взаимной

Труды 12ii Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» – RCDL'2010, Казань, Россия, 2010

изоляции (Isolation) процессов при работе с данными – они не должны мешать друг другу при работе с пересекающимися объектами.

Наряду с ещё одним техническим требованием **долговечности (Durability)** произведённых изменений, отсутствия потери данных после фиксации, первые три условия формируют принципы **ACID** транзакционной модели доступа к данным в конкурирующих транзакциях: Atomic, Consistent, Isolated, Durable. Дальнейшее изложение не предполагает какой-либо одной частной архитектуры вычислительной системы, наличия СУБД и подобных вещей – более удачной является введённая выше абстракция с байтами, которая будет широко использоваться и далее. Однако понятия и подходы, разработанные в рамках теории управления конкурирующими транзакциями, будут привлекаться при дальнейшем анализе. Главной целью данной работы является поиск путей ослабления необходимости в синхронизации конкурирующих процессов.

2 О понятии объекта

Чтобы дальнейшее изложение носило более предметный характер, необходимо остановиться на понятии объекта более детально. Рассмотрим область памяти системы как некоторое пространство байт. Если содержимое одного байта обуславливает правомерность другого, то имеется *направленная зависимость* между этими двумя байтами (чаще она является обоюдной). Если теперь адекватность 3-го байта зависит от содержимого 2-го байта, но не зависит напрямую от содержимого 1-го байта, то имеется также и *нетранзитивная зависимость* 3-го байта от 1-го. Если же содержимое 3-го байта способно потерять адекватность при изменении 1-го равно как и 2-го, то эта зависимость будет *транзитивной*. Наконец, если есть ещё зависимость 1-го байта от 3-го, то имеет место *циклическая зависимость* 1-го, 2-го и 3-го байт.

Более сильные транзитивные и компактные циклические зависимости часто естественным образом соответствуют сущностям предметной области, взятым вместе с их непосредственными связями с ближайшими соседями, а протяжённые области нетранзитивно зависимых данных дополнительно включают все опосредованные межобъектные зависимости. Область транзитивной зависимости назовём **классическим объектом** (со связями), а объемлющую область нетранзитивной зависимости – неклассическим **объектом-замыканием** или замкнутым объектом; если тип зависимости не будет иметь значения, будем говорить просто об объекте. Связи не могут принадлежать одному объекту; вместо этого они разделяются между несколькими объектами, а сами объекты становятся пересекающимися окрестностями в рассматриваемом пространстве байт. Формализацию подобного выделения объектов можно найти в [6]. Если в системе существует множество версий объекта, все они логически рассматриваются входящими в одну окрестность.

Дальнейший анализ предполагает, что сущности предметной области соответствуют одному из двух типов введённых выше объектов или, по крайней мере, заключены внутри них.

3 Постановка проблемы

Поскольку реальные вычислительные узлы не идеальны в смысле надёжности и пропускной способности, в реальных распределённых системах предусматриваются механизмы балансировки нагрузки, тиражирования и восстановления данных, управления и безопасности. К системам обработки данных, как правило, предъявляется ряд требований по обеспечению устойчивости к отказам компонент, постоянной готовности сервиса и синхронности изменений состояния данных.

Все три характеристики являются важными для бесперебойной работы системы, однако оказывается практически невозможным удовлетворить всем трём требованиям в полном объёме. Так, хорошо известно, что требования сильной синхронизации и высокой степени параллельности с конкурирующим доступом являются антагонистами. Сочетать свойства постоянной доступности и толерантности к потерям сообщений внутри системы не менее сложно. Эти проблемы являются широко дискутируемыми в рамках классификации задач, предложенной Брюэром в его CAP-тезисе [4, 5], по которой выделяют требования: согласованности различных версий **Consistency**[1], высокой степени доступности сервиса **Availability** и устойчивости к разделению на части **Partition tolerance**. Тезис гласит, что только две из трёх характеристик могут быть обеспечены проектом, поэтому при абсолютизации требований получаются системы только трёх типов: недостаточно устойчивые к разрывам коммуникаций – **CA**, с возможными отказами обслуживания – **CP** или без синхронного изменения распределённых данных – **AP**.

Применительно к системам обработки данных это означает существование трёх крайних вариантов систем:

- **CA** (неделимая система) – это подход с фрагментацией данных, строгой ACID непротиворечивостью и принципом синхронных изменений с применением двухфазного протокола фиксации транзакций;
- **CP** (отказы обслуживания) – это подход с наличием дублирования, строгой ACID непротиворечивостью и синхронной репликацией изменений с применением метода пессимистических блокировок;
- **AP** (возможна рассогласованность) – это также подход с дублированием, но со слабой, BASE, непротиворечивостью (Eventual consistency) и асинхронной репликацией изменений [1, 8].

Удачно выбранная архитектура распределённой системы обработки данных обычно представляет собой баланс между крайностями, найденный с учётом требований конкретной задачи.

4 Обеспечение устойчивости к отказам

Отказоустойчивость является одной из главных особенностей распределённых систем, и имеется большое количество трудов, посвящённых этой теме [1, 7]. Формат статьи не позволяет раскрыть вопрос вполне, но затронуть его нам необходимо.

Можно выделить два крайних подхода к построению систем распределённой обработки и хранения информации, хотя и допускающих промежуточные варианты. Это:

- фрагментация, или секционирование данных – распределение частей общей базы данных по узлам сети без дублирования информации;
- дублирование, или тиражирование данных – создание тождественных полных копий общей базы данных на каждом узле сети.

Первый подход позволяет минимизировать объём хранимых данных, но предъявляет жёсткие требования к надёжности оборудования и быстродействию каналов передачи информации при выборках. Второй подход позволяет повысить надёжность системы и скорость выборок, но в свою очередь требует больших объёмов дисковой и оперативной памяти для хранения и обработки, а также широких каналов для репликации обновлений между узлами при изменении данных.

Решения типа **CA** основываются на первом подходе – непересекающемся распределении информации по узлам распределённой системы. Именно по этой причине данный тип решений столь чувствителен к разрыву коммуникаций. В то же время два других решения типа **P** основываются на дублировании с дальнейшим синхронным (тип **CP**) или асинхронным (тип **AP**) обновлением информации между узлами системы.

В качестве компромиссного решения, позволяющего оптимизировать требования к производительности и надёжности системы, здесь выгодно применить стратегию размещения данных ближе к потребителю. Эта стратегия сочетает в себе преимущества двух подходов – фрагментации и дублирования информации: часто используемые данные тиражируются и перемещаются ближе к запрашивающим их узлам, что позволяет повысить оперативность запросов на выборку, а редко используемые имеются в единственном экземпляре, что позволяет снизить затраты на хранение и синхронизацию информации при изменениях. В такой среде будут происходить постоянные перемещения данных между узлами (data movement) с целью оптимизации производительности системы в целом, реализуя, таким образом, вариант подсистемы динамической балансировки нагрузки (load balancing).

Как можно заметить, данная стратегия затрагивает третье условие CAP-тезиса – устойчивость системы к разделению на части при отказах узлов или каналов связи, смещая архитектуры **CP** и **AP** типов в сторону типа **CA**. Однако уменьшение надёжности нивелируется ранжированием информации по частоте использования, к тому же оно может быть минимизировано выбором более надёжного оборудования опорных узлов. Таким образом, ослабив одно из условий теоремы, можно получить существенный прирост эффективности распределённой системы в целом.

Следующим логичным шагом в этом направлении является поиск компромисса между типами **CP** и **AP**. К сожалению, это не менее многогранная проблема, и здесь очень многое зависит от типа решаемых задач – насколько свободно они позволяют обрабатывать данные в конкурентном режиме. Способам достижения баланса архитектурного решения между типами **CP** и **AP** посвящён следующий раздел.

Мы начнём со схемы асинхронной репликации в среде с частичным дублированием, а затем посмотрим, какие дополнительные условия синхронизации в какой ситуации нужно на неё наложить, чтобы приблизить её к синхронной схеме репликации.

5 Обеспечение непротиворечивости

В распределённых системах критичной является операция записи данных, когда два процесса меняют взаимозависимые данные или когда один процесс меняет некоторую часть объекта, в то время как второй процесс уже прочитал другую его часть и собирается прочесть изменённую. Такие ситуации называют низкоуровневыми конфликтами совместного доступа к данным, и они находят более или менее удовлетворительное разрешение в рамках подходов по управлению конкурирующими транзакциями [2, 3].

Кроме низкоуровневых конфликтов совместной записи данных в объектных средах с зависимостями могут отдельно возникать логические конфликты, вызванные некоммутативностью операций, выполняемых в конкурирующих процессах. Такие конфликты необходимо выявлять и анализировать отдельно, причём при определённой организации данных может оказаться возможным использовать низкоуровневые коллизии для выявления конфликтов высокого уровня, а высокоуровневую логику – для разрешения ряда конфликтов низкого уровня. Например, для изменения значения поля $\mathbf{x = 0}$ часто можно использовать коммутативные операции: $\mathbf{x = x + 1}$, $\mathbf{x = x + 2}$, результат которых не зависит от порядка выполнения, вместо аналогичных некоммутативных операций: $\mathbf{x = 1}$, $\mathbf{x = 3}$.

В каждой задаче, допускающей параллельность, можно обнаружить области данных, логически не связанные между собой. Эти области данных могут меняться независимо, по крайней мере, на каком-то этапе выполнения задачи, что в точности означает, что операции изменения этих областей данных коммутируют между собой, и здесь степень синхронизации может быть уменьшена. Логику объектов высокого уровня можно использовать для ослабления требований к синхронизации при сохранении непротиворечивости информации, что позволяет применить для распространения изменений прин-

ципы BASE непротиворечивости с асинхронной репликацией, возможно лишь с некоторыми элементами синхронизации. Мы рассмотрим эти шаги более детально с использованием транзакционной модели, но сначала хотелось бы предложить одну реальную распределённую систему, созданную на основе принципов BASE, – это наше обычное информационное общество.

Отдельный человек с технической точки зрения является автономным узлом потребления и производства информации, а средства массовой информации, книги и пресса привносят элементы синхронизации. Каждый индивид не имеет чёткого регламента работы с информацией, он может смотреть телепередачи или нет, быть начитанным или же глубоко невежественным человеком, может быть в курсе событий или не интересоваться новостями – всё это оказывается неважным для прогресса общества в целом. Информация в нём постоянно накапливается и, в общем, не становится противоречивой, хотя каждый отдельный человек обладает только частью всей информации, и она часто бывает неточной или даже искаженной. Здесь можно отметить как значительное дублирование информации: каждый узел имеет в своём личном владении почти все необходимые ему данные, так и наличие системы асинхронной репликации (личное общение, почта, посещение библиотеки, интернет). Кроме того, имеются элементы синхронной репликации, но без гарантии получения сообщений конечным узлом (средства массовой информации). Имея данную модель перед глазами, будет проще сделать ряд обоснованных предположений в дальнейшем.

5.1 Многоверсионная транзакционная модель

Методы анализа, разработанные в рамках подходов к управлению конкурирующими транзакциями, могут быть с успехом применены и при проектировании системы репликации данных, если считать началом транзакции момент актуализации узла данными других серверов, а её завершением – момент обновления серверов данными текущего узла. Все данные, используемые транзакцией, должны быть согласованы между собой – в случае наличия частичных изменений в пределах системы эти данные узла необходимо обновить. При таком соответствии синхронизация в распределённой системе может быть точно описана транзакционной моделью совместного доступа к данным.

Рассмотрим две транзакции, которые читают и пишут некоторые данные. Возможны следующие пересечения этих транзакций по данным и по времени (рис. 1).

Для устранения конфликтов «чтение-запись» можно применить режим изоляции Guaranteed view (гарантированное представление), когда читающая транзакция видит моментальный снимок – состояние БД на момент своего старта. Для минимизации более серьёзных конфликтов «запись-запись» можно основать систему репликации на принципах многоверсионности баз данных. При этом единственными допустимыми операциями являются добавление новых или пометка удаляемых записей, никаких реальных операций удаления или модификации записей БД производиться не должно.

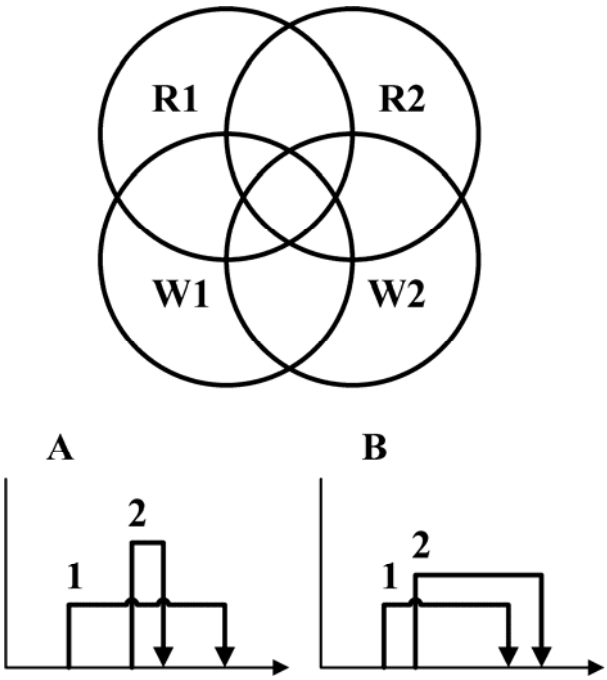

Рис. 1. Виды пересечения транзакций по данным и по времени. Обозначения: R1, R2 – область чтения, W1, W2 – область записи 1-й и 2-й транзакций; A, B – варианты пересечения по времени

При уровне изоляции Guaranteed view области чтения можно не рассматривать в низкоуровневой модели, поскольку после своего старта транзакция не видит никаких изменений данных, даже подтверждённых другими транзакциями. На физическом уровне достаточно рассмотреть состояние в области пересечения по данным **W1 * W2** и времени **A** или **B**, то есть всего два варианта конфликтов: **W1 * W2 * A** и **W1 * W2 * B**. Обе эти ситуации рассматриваются как конфликты записи, при этом одна из транзакций откатывается, иначе может возникнуть противоречивость данных, поскольку данные области **W1 * W2** могут быть взаимозависимы с данными как области **W1**, так и области **W2**, и после изменений первой транзакции результаты второй транзакции могут потерять согласованность с остальной частью БД.

Ситуация существенно усложняется при наличии высокоуровневых логических конфликтов. Рассмотрим область пересечения по данным **R1 * R2 * W1 \ W2**, менять которую может только транзакция **tr1**. Здесь никакого конфликта в транзакционной модели не возникает, и, тем не менее, результаты транзакции **tr2** могут оказаться противоречивыми, поскольку она частично использует устаревшие данные из области записи **W1** транзакции **tr1**. Использование режима изоляции транзакций Read Committed (видимость подтверждённых изменений) не способно улучшить ситуацию, по-

скольку транзакции **tr2** пришлось бы постоянно контролировать неизменность уже прочитанных ею данных.

Кроме того, уже реализация режима изоляции Guaranteed view, обеспечивающего временной срез данных для устранения конфликтов «чтение-запись», в распределённой системе сама по себе является непростой задачей, требующей синхронизации изменений версий объекта.

5.2 Тотальный контроль непротиворечивости

Для решения данных проблем можно было бы применить оптимистические методы управления транзакциями с обратной валидацией, выполняя пост-проверку того, что результат транзакции согласуется с результатами других транзакций, зафиксированных после её начала. Такую проверку можно сделать либо на логическом уровне, контролируя непротиворечивость сохраняемых объектов, либо на физическом уровне, контролируя неизменность всех данных, использованных транзакцией **tr2** в своей работе.

Первый вариант очень интересен, поскольку практически не требует синхронизации процессов при записи, но, к сожалению, трудно реализуем – он требует развитой логики процедур проверки, которым необходимо иметь априорную информацию обо всех ограничениях предметной области.

В свою очередь, при втором варианте возникают общие сложности синхронизации изменений во всей распределённой структуре, а также проблема учёта новых записей, созданных транзакцией **tr1** за время работы **tr2**. Такие записи не были учтены транзакцией **tr2**, хотя могли бы повлиять на результат. В общем случае для оценки, какие из новых записей могли оказать такое влияние, придётся повторить выполнение транзакции **tr2** с самого начала при любом добавлении записей в БД, что приводит к запрету всякого параллелизма вообще.

5.3 Отказ от контроля физических конфликтов

В качестве альтернативы тотальному контролю непротиворечивости можно предложить вариант без отслеживания физических конфликтов вообще. Поскольку мы не в состоянии обеспечить непротиворечивость данных во всех случаях, не имеет особого смысла заботиться о ней и в некоторых из них. Вместо этого необходимо спроектировать высокоуровневую обработку информации таким образом, чтобы она была невосприимчивой к возникающей частичной рассогласованности данных или на логическом уровне не допускала её вовсе.

Такой подход отклоняется от транзакционной модели, здесь не будет валидации на физическом уровне – транзакции всегда будут завершаться. Механизмы многоверсионности обеспечат возможность хранения любых взаимоисключающих изменений: одна транзакция сможет удалить (пометить к удалению) запись, в то время как другая будет её менять, и обе потом успешно подтвердят свои изменения. По этой причине необходимо рассмотреть варианты пересечения транзакций по данным и по времени более тщательно.

Прежде всего, следует обеспечить атомарность объектов, которая гарантирует, что никакой объект не будет зафиксирован в системе лишь частично. При сложной организации объекты могут храниться в нескольких местах БД, поэтому в отсутствие контроля физических конфликтов гарантии атомарности необходимо предусматривать дополнительно. Этого можно добиться двумя разными способами: либо предусмотрев элементы синхронизации, запрещая изменение всех данных объекта конфликтующей транзакцией, либо полностью перезаписывая в каждой транзакции всю область данных объекта (и всех пересекающихся с ним – в случае классических объектов), тогда эту область можно будет рассматривать как некую атомарную логическую запись базы данных. К сожалению, этот последний вариант неприемлем для сильно связанных данных – в таком случае может оказаться невозможным выделение замкнутой окрестности без пересечений с другими объектами.

5.4 Отказ от контроля логических конфликтов

Проанализируем теперь коллизии «чтение-запись» и «запись-запись» – последние целиком перенеслись с низкого на высокий уровень из-за отказа от контроля конфликтов на физическом уровне. Оба типа коллизий можно рассматривать единообразно, поскольку они имеют одинаковую причину: результат транзакции основывается на данных, полностью или частично потерявших актуальность за время выполнения этой транзакции. Такие коллизии не приводят к проблемам, если результат транзакции также оказывается устаревшим и полностью перекрывается более свежими версиями объектов, или даже если устаревший результат полностью перекрывает все зависимые более свежие данные – тогда состояние данных просто откатится немного назад.

В противном случае, когда последняя версия данных будет содержать в какой-то своей части неактуальные результаты, информация может стать противоречивой. Хотя это необязательно, поскольку частный результат может оказаться совместимым с произведёнными другой транзакцией изменениями данных в области зависимости. Кроме того, само по себе смешение результатов различной актуальности – несогласованность – ещё не обязательно приводит к противоречивости информации, поскольку даже сама входящая в систему информация практически никогда не синхронизирована по времени и допускает различный порядок следования событий! При этом некоммутативность операций с данными становится несущественной, поскольку полностью вуалируется естественной неупорядоченностью этих операций.

Обратимся опять к рис. 1. Пусть первая транзакция **tr1** работает со своим *неклассическим* объектом-замыканием **R1 + W1**, а вторая **tr2** – со своим

R2 + W2. Хотя неклассические объекты пересекаются, каждый из них меняется непротиворечиво для себя и для других на основании только тех данных, которые находятся в принадлежащих ему областях (например, для первого объекта это **R1 + W1**) – в противном случае границы областей, занимаемых объектами-замыканиями были бы другими. Поскольку входящая информация неупорядочена, а транзакции запускаются на основе каких-то внешних событий, то нет никаких гарантий того, что транзакция **tr1** будет использовать свой объект до изменения части области данных **W1 * W2** транзакцией **tr2** и наоборот. Поэтому состояние этой области может быть тем или другим или даже измениться за время выполнения транзакции – всё это никак не может привести к противоречивости информации в БД, хотя в ней и могут оказаться не самые последние данные, если последняя из завершившихся транзакций выполнялась длительное время.

Таким образом, отличие технической несогласованности, вызванной параллельностью обработки, от естественной, вызванной отсутствием предопределённого порядка входящих событий, состоит только в том, что в первом случае итоговое состояние объекта-замыкания может представлять собой некоторую комбинацию различных состояний, логически несовместимых между собой (если их допускает структура хранения данных). Это последствия нарушения атомарности, и если средства её поддержки были предусмотрены, то *техническая и естественная несогласованности абсолютно неразличимы, и никакой дополнительной синхронизации не требуется.*

5.5 Добавление элементов синхронизации

К сожалению, в отличие от неклассических объектов, гарантия атомарности на уровне *классических* объектов не является достаточным условием обеспечения непротиворечивости БД. Случай опосредованно, нетранзитивно зависимых по некоммутативным операциям областей данных не позволяет в полной мере обеспечить атомарность на этом уровне – нетранзитивные зависимости представляют собой межобъектные логические зависимости и не охватываются классической объектной моделью.

Например, пусть имеются две области данных, представляющие непересекающееся сущности предметной области: **Xp = A + D** и **Yp = B + E**, где **C = D + E** является объединением сфер взаимного влияния, сфер действия связей. Этим сущностям будут соответствовать пересекающиеся классические объекты со связями **X = A + C** и **Y = B + C**.

Допустим, с небольшим интервалом над **X** стартовали две транзакции, одна из которых завершилась и перевела объект в состояние **X1 = A1 + C1**. Затем стартовала третья транзакция над объектом **Y1 = B + C1** и перевела его в состояние **Y3 = B3 + C3**. Потом завершилась вторая транзакция, перекрыв результаты первой: **X2 = A2 + C2**. В итоге объект **Y3** оказался в состоянии **Y3 = B3 + C2**, что является потенциально опасным в случае изменения областей данных как классических объектов, без учёта окружения. Заметим, что если бы рассмотренным сущностям соответствовали неклассические объекты, то никаких вопросов к непротиворечивости не возникло. Проблема в том, что таких отдельных объектов-замыканий может не существовать из-за взаимных зависимостей данных, тогда пример сведётся к тривиальному случаю: **X = Y = A + B + C**. В классическом случае второй транзакции нельзя было завершаться, поскольку её данные уже были изменены другой транзакцией.

Для синхронизации в данном случае можно применить архитектуру «доска объявлений». По узлам системы распределяются доски объявлений, на которых помещаются списки классических объектов, включая их непосредственные связи (область **X = A + C** транзитивной зависимости в примере, рассмотренном выше). Каждый объект имеет свой домашний узел или группу узлов приписки, так что системе всегда известно, где искать соответствующую доску. В процессе запроса данных для предотвращения конфликтов «чтение – запись» проверяется согласованность версий пересекающихся областей данных, а при записи контролируется их неизменность другими транзакциями, предотвращая порчу данных при конфликтах «запись – запись» – в этих случаях транзакция откатывается, либо производятся иные действия по акцепту изменений.

Такая архитектура обеспечивает строгую ACID непротиворечивость при наличии множества распределённых копий данных, но за это требует проверки каждого читаемого объекта по доске объявлений. Если задача допускает использование не самых актуальных версий, то быстродействие системы может быть существенно увеличено с переходом на принципы BASE – тогда обращений к доскам при чтении не требуется. Действительно, пусть транзакция желает основываться на данных годичной давности, тогда, очевидно, она может смело запрашивать срез данных того периода, не опасаясь, что какие-то объекты прочитаются неатомарно – ведь все транзакции, начатые в прошлом году, уже давно завершены, и те версии данных уже никем не могут быть изменены. Ввод не такой большой, но разумной толерантности к актуальности информации в запросах данных позволяет реализовать BASE-принципы без какой-либо существенной перестройки системы. Задержка определяется характерным временем асинхронной репликации данных между узлами системы.

Толерантные к актуальности данных задачи нередки – это все темы, не относящиеся к задачам реального времени. Естественная задержка поступления новой информации в систему маскирует техническую дельту времени в запросах точно так же, как отсутствие предустановленного порядка событий вуалирует некоммутативность операций с данными! Это значит, что здесь мы получаем такой же значимый выигрыш: кроме асинхронной репликации *для обеспечения атомарности чтения никакой дополнительной синхронизации не требуется.*

5.6 Дальнейшие шаги по оптимизации

Если сущности предметной области можно рассматривать как слабо связанные объекты, то подход без контроля конфликтов будет работать и *без глобальной синхронизации записи*. Но даже при наличии нетранзитивных зависимостей имеются задачи, не чувствительные к наличию некоторого объёма противоречий в данных; главное – чтобы он не нарастал со временем. Такого поведения можно ожидать от моделей, обладающих чертами эргодических цепей Маркова, когда поведение случайного процесса перестаёт определяться его начальным состоянием. В этот ряд попадают также задачи по интенсивной обработке информации, содержащей большое количество зависимостей с обратными связями. Для реализации данного подхода в распределённой системе нужно лишь убрать все обращения к доскам объявлений; несмотря на утрату атомарности записи (а возможно – и чтения) объекта как совокупности версий, в данном случае необходимо ставить вопрос о сохранении атомарности записи и чтения каждой отдельной его копии.

В пространстве байт можно ввести ещё одно поле – продукций, если мы имеем возможность разделить по времени процессы обновления различных областей. В отличие от статического поля зависимостей, использованного для выделения объектов, поле продукций является динамическим. Все области будут упорядочены векторным полем потоков данных, возможно, содержащем циклы. Тогда базу данных можно рассматривать как самоорганизующуюся нелинейную динамическую систему, в которой потоки данных, в ходе эволюции стремясь к аттрактору, постепенно затирают ошибочные данные. Насколько реальным будет такой эффект – интересный вопрос, также зависящий от задачи, но само существование нашего общества индивидов доказывает, что в принципе это возможно.

Заключение

Представленные в работе понятия и методы, касающиеся выбора оптимальной стратегии обеспечения непротиворечивости данных, могут найти своё применение при проектировании распределённых систем различного назначения. Особый интерес для упрощения анализа, как представляется, имеет концепция «пересекающихся объектов» – окрестностей в пространстве байт, включающих области зависимостей данных. Такой «топологический» подход к анализу хранилищ данных в совокупности с методами упомянутой теории динамических систем, вероятно, способен привести к новым интересным результатам в дальнейшем: предтопологические пространства уже сейчас используются в работах, связанных с анализом сложных структур [6].

Разработанные методы и архитектура на базе «доски объявлений» используются в работе над проектом глобального ресурса знаний в системе извлечения информации из текстов ИСИДА-Т.

Литература

[1] Таненбаум Э., ван Стеен М. Распределённые системы. Принципы и парадигмы. – СПб.: Питер, 2003. – ISBN: 5-272-00053-6.

[2] Чардин П. Многоверсионность данных и управление параллельными транзакциями// «Открытые системы». – 2005. – № 1. – С. 64-69.

[3] Bretl R. Achieving high concurrency in object oriented databases. – ODBMS.ORG, 2006. – http://citforum.ru/database/articles/bretl.

[4] Brewer E.A. Towards robust distributed systems (abstract)//Proc. of the 19th Annual ACM Symposium on Principles of Distributed Computing, Portland, Oregon, United States, 2000, July 16 – 19. – P. 7.

[5] Gilbert S., Lynch N.A. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services// SIGACT News. – 2002. – V. 33, No 2. – P. 51-59.

[6] Largeron C., Bonnevay S. A pretopological approach for structural analysis// Information Sciences. – 2002. – V. 144, July. – P. 169-185.

[7] Lynch N.A. Distributed algorithms. – Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1996. – ISBN 1-55860-348-4.

[8] Vogels W. Eventually consistent// ACM Queue. – 2008. – V. 6, No 6. – P. 14-19.

Design of distributed systems of processing object data structures

A.A. Demidov

This paper is devoted to the problem of reliability and consistency of data in concurrent transactions in distributed system. Some of well-known principles of distributed system architecture design are reinvestigated to discover advantages of their usage in order of resolving problems of given specific tasks. Emphasis is made on the problem of choice the optimal strategy of providing consistency.

♣ Работа выполнена по программе фундаментальных исследований Президиума РАН №3, проект «Высокопроизводительные масштабируемые средства работы с фактографическими базами большого объёма»

[1] Это другая Consistency: «согласованность» в отличие от «непротиворечивости» в ACID, хотя эти понятия связаны: при различии версий обеспечить непротиворечивость в рамках полной системы становится проблематично