BAKI UNİVERSİTETİNİN XƏBƏRLƏRİ

№3 Fizika-riyaziyyat elmləri seriyası 2011

İNFORMATİKA

УДК 681.324

РАЗРАБОТКА АЛГОРИТМА СИНХРОНИЗАЦИИ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ ОБРАБОТКИ ИНФОРМАЦИИ

А.А.АЛИЕВ, Х.А.ГАСАНОВ, Э.Н.ИСРАФИЛОВА
Бакинский Государственный Университет
aaliyev@mail.ru , xalid.h@gmail.com

В данной работе исследуется проблема синхронизации прикладных процессов в распределенных системах обработки информации. Предлагается модель распределенной системы и на основе этой модели рассматривается транзактная обработка информации в распределенных базах данных. Разработан алгоритм синхронизации в распределенных системах. Преимуществами этого алгоритма являются обеспечение сериализуемости параллельно выполняемых транзакций, целостности распределенной базы данных и отсутствие рестартов транзакций.

Ключевые слова*: *распределенные системы, транзакция, синхронизация процессов, целостность, распределенная база данных, сериализуемость

Распределенные системы обработки информации, или просто распределенные системы (РС), представляют собой множество территориально отдаленных друг от друга узлов, объединенных системой передачи данных и взаимодействующих посредством обмена сообщениями. Одной из проблем, возникающих в теории и практике РС, функционирующих в сетевой среде, является синхронизация процессов, протекающих в узлах РС. К настоящему времени теоретические проблемы и вопросы реализации синхронизационных процессов в РС исследуются многими учеными и являются актуальными. Синхронизация необходима процессам для организации совместного использования ресурсов. Под синхронизацией понимается скоординированное корректное управление процессами, протекающими в РС как последовательно, так и параллельно, и конкурирующими между собой за использование некоторого ресурса. Одним из наиболее развитых видов РС являются системы распределённой базы данных (РБД) [1].

Рассмотрим одну из задач, в основе которой лежит проблема синхронизации процессов, а именно, задачу обеспечения сериализуемости параллельно выполняемых транзакций в РС. Эта проблема обсуждается в литературе в течение последних лет. На данном этапе она также актуальна.

Транзактная обработка информации. В настоящее время общепринятым методом обработки информации в распределенных системах является транзактная обработка. Это означает, что весь процесс обработки состоит из множества заданий, называемых транзакциями. Транзакции, т.е. единичные работы, должны программироваться и выполняться согласно определенным правилам.

Примем следующую модель РС, реализующую транзактную обработку. РС является совокупностью узлов $S = \{S_i / \overline{i = 1,n}\}$ и системы передачи данных, связывающей любую пару узлов. Во множестве узлов S в каждый момент времени функционируют два подмножества: $\{TM_i\}$ – подмножество узлов, инициирующих работы (транзакции); $\{DM^k\}$ – подмножество узлов, выполняющих (обрабатывающих) транзакции. Транзакции, инициированные узлом TM_i, обозначим через T_i. В общем случае T_i выполняется в нескольких узлах DM^k. Ту часть T_i, которая выполняется в одном узле DM^k, назовем подтранзакцией и обозначим T_i^k. Будем отождествлять транзакцию с множеством составляющих ее подтранзакций: $T = \{T_i^1, ..., T_i^{n_i}\}$.

Каждая подтранзакция T_i^k перед началом своей работы должна захватить в узле DM^k локальную базу данных, которую назовем информационным ресурсом данного узла, или просто ресурсом.

Подтранзакции одной и той же транзакции, а также разных транзакций в различных узлах DM^k могут выполняться параллельно. Ввиду случайности и независимости процесса инициации транзакций узлами TM_i в узлах DM^k могут образовываться очереди подтранзакций, ждущих обработки. Совокупность всех очередей назовем распределенным планом выполнения подтранзакций (РПП). Координация параллельной обработки транзакций в рамках всей РС проводится системой управления транзакциями.

Рассмотрим процесс параллельной и независимой инициации транзакций со стороны TM_i и установления, составляющих эти транзакции подтранзакций в локальной очереди в узле DM_i^k на следующем примере. Пусть каждые из шести узлов $TM_1 - TM_6$ инициируют по одной транзакции на множестве из трех узлов $DM^1 - DM^3$:

$$T_1 = \{ T_1^3 \}, \quad T_2 = \{ T_2^1, T_2^2 \},$$
$$T_3 = \{ T_3^1, T_3^3 \}, \quad T_4 = \{ T_4^2, T_4^3 \}, \qquad (1)$$

$T_5 = \left\{ T_5^1, T_5^2, T_5^3 \right\}$, $T_6 = \left\{ T_6^1 \right\}$.

На время допустим, что формирование очередей подтранзакций узлом DM^k происходит в режиме FIFO. При этом никаких предположений относительно порядка инициации транзакции не делается. В связи с различными транспортными задержками при доставке сообщений об инициации подтранзакций из узлов $TM_i \left(i = \overline{1,6} \right)$ в узлы $DM^k \left(k = \overline{1,3} \right)$ упорядоченность очередей, образовавшихся в узле DM^k, носит, в общем случае, непредсказуемый характер.

Пусть в узлах DM^k образовались очереди $Q^k \left(k = \overline{1,3} \right)$, показанные на рис.1. Построим, так называемый, Q - граф, соответствующий этим очередям, т.е. множеством вершин этого графа будет множество всех подтранзакций T_i^k, стоящих в очереди. Дугами соединяются те пары вершин, которые попали в общую очередь и являются соседями в ней. Направление (ориентацию) дуги примем согласно следующему правилу: дуга направлена от некоторой подтранзакции, стоящей в очереди, к подтранзакции, стоящей непосредственно перед ней. Получим орграф, показанный на рис.2. Q - граф в общем случае несвязный. Максимально он имеет столько компонентов связности, сколько узлов DM^k имеется в РС. Q - граф – это лес, каждое дерево которого имеет очень простой вид. Каждый компонент связности Q - графа называется локальным планом выполнения подтранзакций (ЛПП) для соответствующего узла DM^k.

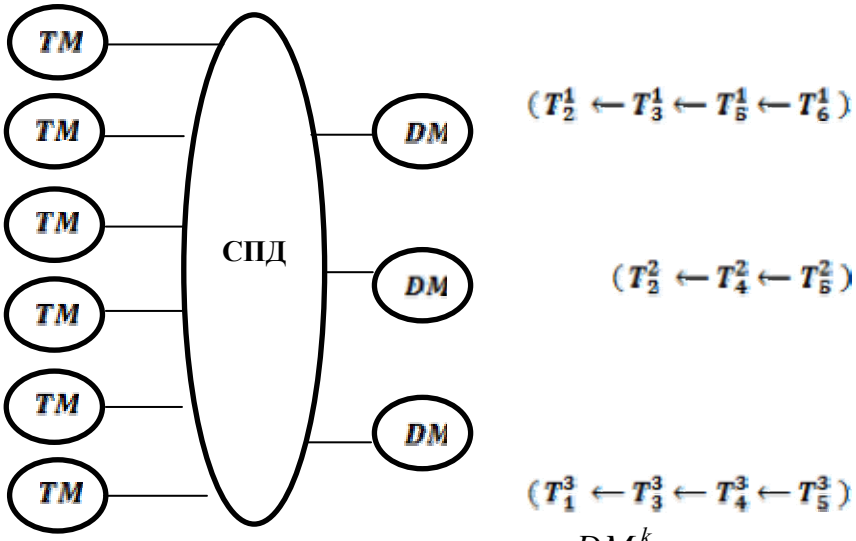

Рис. 1. Очереди, образовавшиеся в узлах DM^k, совокупность их является исходным РПП

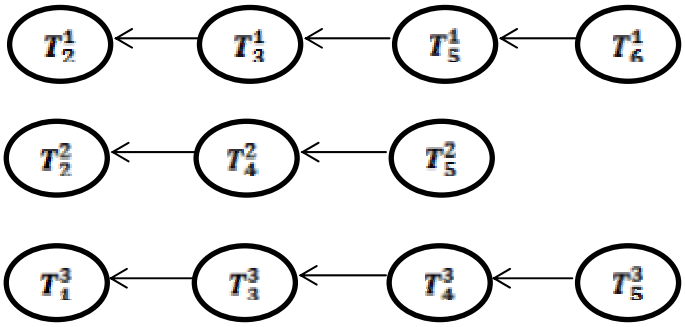

Рис.2. Q - граф, соответствующий очередям на рис.1.

Теперь построим так называемый, D - граф (граф зависимостей). Для этого склеим вершины Q - графа, имеющие одинаковые нижние индексы, и опустим верхние индексы (рис.3.). Число вершин D - графа равно числу узлов TM_i, а число дуг вычисляется по формуле:

$$N = \sum_{k=1}^{n_{DM}} \left(l^k - 1 \right), \qquad (2)$$

где l^k – длина очереди в DM^k. Если D - граф является деревом или лесом, т.е. множеством несвязных компонентов - деревьев, то его можно описать множеством упорядоченных конечных последовательностей, при этом наличие между некоторыми парами нескольких (однонаправленных) дуг во внимание не принимается. Так, *D*- граф на рис.3. можно описать тремя последовательностями:

$$\begin{aligned} &T_2 \leftarrow T_3 \leftarrow T_5 \leftarrow T_6, \\ &T_2 \leftarrow T_4 \ , \qquad (3) \\ &T_1 \leftarrow T_3 \leftarrow T_4 \leftarrow T_5, \end{aligned}$$

где $\leftarrow$ – знак предшествования. Эти последовательности описывают имеющуюся на *D*- графе (введенную на множестве вершин) частичную упорядоченность вершин. Используя транзитивность отношения предшествования можно также утверждать, что, например, $T_1 \leftarrow T_4$ или $T_2 \leftarrow T_5$. Вместо с тем в силу только частичной упорядоченности отношение предшествования для вершин T_1 и T_2 не определено.

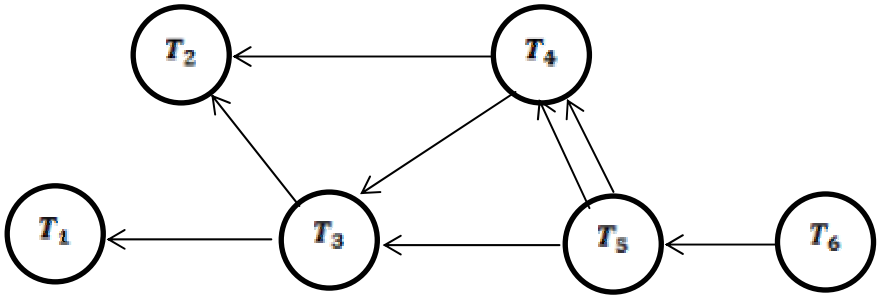

Рис.3. D - граф, соответствующий очередям на рис.2.

Если D - граф не является деревом или лесом, т.е. один из его компонентов связности содержит циклы, то ввести частичную упорядоченность на множестве его вершин, инцидентных некоторому циклу, очевидно, невозможно. На рис.4. приведен пример построения *D*- графа с циклом для транзакций:

$$\begin{aligned} T_1 &= \left\{ T_1^1, T_1^2 \right\}, \\ T_2 &= \left\{ T_2^1, T_2^3 \right\}, \\ T_3 &= \left\{ T_3^2, T_3^3 \right\}. \end{aligned} \qquad (4)$$

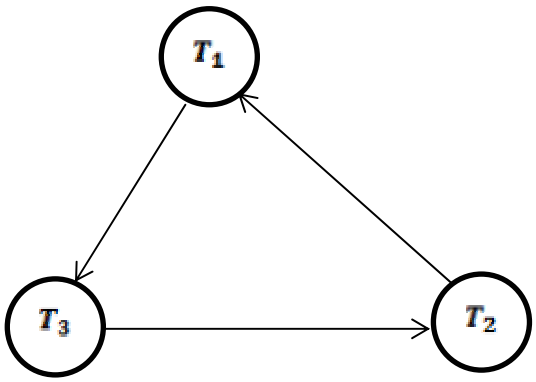

Рис.4. *D*- граф с циклом

Что означает частичная упорядоченность на *D*- графе? Рассмотрим вновь записанные выше последовательности (3). Первая последовательность означает, что все подтранзакции T_5, запланированы к выполнению раньше, чем таковые для T_6, т.е. можно сказать, что вся транзакция T_5 запланирована к выполнению в РПП позже, чем T_2. Аналогично

T_4 запланирована к выполнению в РПП позже, чем T_2, и т.д. Транзакция T_3 выполняется раньше, чем T_5, однако T_1 не конфликтует с T_2.

Пользуясь некоторой свободой действий, можно три последовательности (3) слить в одну так, чтобы отношение предшествования распространялось на все транзакции, например:

$$T_2 \leftarrow T_1 \leftarrow T_3 \leftarrow T_4 \leftarrow T_5 \leftarrow T_6 \qquad (5)$$

Этим мы ввели глобальную упорядоченность транзакций, которая не нарушает ни одной из отмеченных выше частичных упорядоченностей.

В принципе, если отвлечься от требования высокой производительности РС, т.е. не использовать возможности параллельного выполнения транзакций, можно было бы выполнять транзакции согласно определяемому (5). Это означает, что подтранзакция выполняются не в соответствии с очередями FIFO в исходном РПП, а вначале выполняются все подтранзакции, порожденные транзакцией T_2, затем – порожденные транзакцией T_5 и т.д.

Сохранение отношения предшествования означает, что последовательное выполнение транзакций в таком порядке приведет все локальные базы данных к тому же состоянию, что и выполнение с параллелизмом согласно последовательностям подтранзакций, определяемым очередями на рис.1. То есть исходный РПП с тремя параллельными ветвями, приведенный на рис.1, эквивалентен последовательному плану, определяемому (5). Следует отметить, что, не нарушая частичных упорядоченность, глобальную упорядоченность транзакций можно ввести различными способами, например:

$$T_2 \leftarrow T_4 \leftarrow T_6, \qquad (6)$$

$$T_2 \leftarrow T_1 \leftarrow T_3 \leftarrow T_5 , \qquad (7)$$

$$T_3 \leftarrow T_4 \leftarrow T_5 \leftarrow T_6. \qquad (8)$$

Поскольку каждый из последовательных планов (6) – (8) эквивалентен исходному РПП, они эквивалентны друг другу. Таким образом, некоторый РПП называется сериализуемым, если он эквивалентен, по крайней мере, одному последовательному (сериальному) плану.

Признаком сериализуемости РПП является ацикличность соответствующего D - графа. Важность класса сериализуемых РПП вытекает из следующих рассуждений. Каждая транзакция программируется так, что-

бы, начав работать над целостной РБД, она оставляла после себя также целостную РБД. Из этого непосредственно вытекает, что любой последовательный план так же не нарушает целостности РБД. Учитывая тот факт, что сериализуемый РПП приводит РБД к тому же состоянию, что и некоторый последовательный, можно утверждать, что сериализуемый план не нарушает целостность РБД [2].

Вернемся еще раз к неоднозначности глобальной упорядоченности множества транзакций. Для доказательства сериализуемости РПП достаточно доказать его эквивалентность хотя бы одному последовательному плану. Наличие нескольких последовательных планов, эквивалентных исходному, ничего нового не дает. При этом следует отметить, что различные последовательные планы эквивалентные исходному, приводят РБД к одному и тому же конечному состоянию через различные промежуточные. При одном и том же исходном множестве транзакций, но при различных транспортных задержках могут возникать различные сериализуемые РПП и, как следствие, различные соответствующие последовательные планы, приводящие в общем случае к различным конечным состояниям РБД. Нельзя судить, какое из этих состояний правильное, какое неправильное, можно утверждать лишь одно: все они сохраняют целостность РБД. Указанные выше промежуточные состояния РБД в общем случае нецелостны, что и является основной причиной принятого принципа атомарности транзакций.

Одним из методов обеспечения сериализуемости параллельно выполняемых транзакций является механизм временных меток[3]. Временная метка – эта уникальное число, которое присваивается транзакции. В РС каждый узел имеет свой собственный уникальный идентификационный номер. Упорядочение по временным меткам – это подход, посредством которого последовательность выполнения транзакций подчиняется некоторой приоритетной дисциплине, в соответствии с их временными метками. Временные метки присваиваются на операции чтения (Read) и записи (Write) данной транзакции. Две транзакции (операции) называются конфликтующими, если они оперируют с одним и тем же ресурсом и, по крайней мере, одна из операция Write. Такие конфликтные ситуации называются Read-Write (RW) – конфликтами и Write -Write (WW) – конфликтами. Таким образом, необходимы механизмы, синхронизирующие конфликтующие операции, с учетом их временных меток. Такие механизмы носят названия RW-синхронизации и WW-синхронизации, соответственно, для одноименных конфликтов.

Ниже приводится алгоритм синхронизации параллельно выполняемых транзакций в распределенных системах обработки информации.

Алгоритм.

Шаг 1. Узел TM_i инициирует транзакцию, присваивает ей вре-

менную метку *ts,* которая образуется конкатенацией значений локального времени и идентификационного номера узла и отправляет $ts(T_i^k)$ соответствующим узлам DM^k. При этом временные метки переносятся на все операции чтения *(Read)* и записи *(Write*) данной транзакции и обозначаются $ts(R)$ и $ts(W)$, соответственно.

Шаг 2. Каждый узел DM^k получив эти операции, помещает их в очередь (буфер) и запоминает имя каждого узла, приславшего данную операцию. Все операции помещаются в очередь в строгом порядке в соответствие их временным меткам. Обозначим через $ts(R_{min})$ минимальную временную метку любой буферизованной *Read* операции, а через $ts(W_{min})$ - *Write* операции.

Шаг 3. Каждый DM^k ведет учет в виде таблицы временных меток всех TM_i.

Шаг 4. Для *Read-Write (RW)* синхронизации:

а) операция *Read* передается на выполнение только в том случае, если, с одной стороны, DM^k буферизовал *Write* операции от каждого узла TM_i, а с другой стороны, выполняется условие $ts(R) < ts(W_{min})$;

б) операция *Write* выполняется, если, с одной стороны, DM^k буферизовал *Read* операции от каждого узла TM_i, а с другой стороны, выполняется условие $ts(W) < ts(R_{min})$.

Шаг 5. Для *Write -Write (WW)* синхронизации операция *Write* передается на выполнение в том случае, если, с одной стороны, DM^k буферизовал *Write* операции от каждого узла TM_i, а с другой стороны, выполняется условие $ts(W) < ts(W_{min})$. Отметим, что при такой синхронизации те операции *Write,* которые не конфликтуют между собой могут выполняться не соответствуя временным меткам.

Шаг 6. Каждый узел DM^k перед тем как передать операцию на выполнение, отправляет управляющее сообщение только тем узлам TM_i, для которых отсутствует информация в таблице временных меток и ждет не дольше, чем до момента времени

$$T = 2 * T_{max}^{dist} + \delta,$$

где T_{max}^{dist} - максимальное время прохождения сообщения между любыми двумя узлами; δ - максимальное время для обработки управляющего сообщения.

Единственной целью такого сообщения является передача вре-

менной метки, в DM^k, которую он ожидает от каждого узла TM_i.

Шаг 7. После того как *Read* или *Write* операции выставляются на выполнение, значения $ts(R_{min})$ и $ts(W_{min})$ в узлах DM^k изменяются.

Утверждение. Предложенный алгоритм синхронизации обеспечивает сериализуемость параллельно выполняемых транзакций в распределенных системах обработки информации, сохраняет целостность распределенной базы данных и корректен.

Заключение. Таким образом, в работе рассматривается проблема синхронизации прикладных процессов в распределенных системах. Анализируется одна из задач, в основе которой лежит проблема синхронизации процессов, а именно, задача обеспечения сериализуемости параллельно выполняемых транзакций в распределенных системах. Приводится алгоритм, обеспечивающий сериализуемость параллельно выполняемых транзакций и целостность распределенной базы данных. Преимуществом данного алгоритма заключается в том, что отсутствует рестарты транзакций.

ЛИТЕРАТУРА

1. M.T.Ozsu, P.Valduries. Principles of Distributed Database Systems, Prentice-Hall, 1999, p.666.
2. Schuld H., Alonso G., Beeri C. Atomicity and isolation for transactional processes // ACM trans. on data. sys., 2002, v.27, № 1, p.63-116.
3. Baldoni R. Melideo G. On the minimal information to encode timestamp in distributed computations // Information processing letters, 2002, v.84, № 3, p.159-166.

İNFORMASİYANIN İŞLƏNMƏSİNİN PAYLANMIŞ SİSTEMLƏRİNDƏ SİNXRONLAŞMA ALQORİTMLƏRİNİN İŞLƏNMƏSİ

Ə.Ə.ƏLİYEV, X.A.HƏSƏNOV, E.N.İSRAFİLOVA

XÜLASƏ

Məqalədə informasiyanın işlənməsinin paylanmış sistemlərində tətbiqi proseslərin sinxronlaşması problemi tədqiq olunur. Paylanmış sistemin modeli təklif olunur və bu model əsasında paylanmış verilənlər bazasında informasiyanın tranzakt emalına baxılır. Paylanmış sistemlərdə sinxronlaşdırma alqoritmi təklif olunmuşdur. Bu alqoritmin üstün cəhətləri paralel yerinə yetirilən tranzaksiyaların seriallanması, paylanmış verilənlər bazasının tamlığının təmin olunması və tranzaksiyaların restartlarının olmamasıdır.

Açar sözlər: paylanmış sistemlər, tranzaksiya, proseslərin sinxronlaşdırılması, tamlıq, paylanmış verilənlər bazası, seriallanabilmə.

DEVELOPMENT OF SYNCHRONIZATION ALGORITHMS IN DISTRIBUTED SYSTEMS OF INFORMATION PROCESSING

A.A.ALIYEV, Kh.A.HASANOV, E.N.ISRAFILOVA

SUMMARY

In this paper the synchronization problem of applied processes in the distributed systems of information processing is investigated. A model for the distributed systems is given and on the basis of this model, transact processing of information in the distributed system is considered. A synchronization algorithm is given in the distributed systems. The positive sides of this algorithm are serialization of parallel transactions, integrity of the distributed database, and absence of restarts of the transactions.

Keywords: distributed systems, transaction, synchronization of processes, integrity, distributed database, serializability.

Поступила в редакцию: 15.09.2011 г.
Принято к печати: 03.10.2011 г.