СИНХРОНИЗАЦИЯ КЭШИРУЕМЫХ БАЗ ДАННЫХ МОБИЛЬНЫХ РАБОЧИХ МЕСТ

А.М. Полоневич, М.П. Ревотюк

Белорусский государственный университет информатики и радиоэлектроники,
кафедра информационных технологий автоматизированных систем
6, П.Бровки, г. Минск, Беларусь
телефон: + 375(17)293-86-58; e-mail: rmp@bsuir.by

Рассматривается схема повышения оперативности доступа к базам данных пользователями мобильных рабочих мест, обладающих возможностью кэширования используемых данных.

Ключевые слова – распределенные системы управления, базы данных, синхронизация баз данных .

ВВЕДЕНИЕ

При использовании большинства систем управления базами данных (СУБД) стандартно поддерживается функция репликации баз между серверами, но установка СУБД на специализированный аппаратный комплекс или ноутбук мобильного пользователя часто абсолютно неприемлема. При этом компьютер пользователя пригоден для сохранения полного объема потенциально используемых данных – кэширования базы, но может по разным причинам отключаться от сети (других пользователей) на неопределенное время.

Предмет рассмотрения – одноранговые системы обработки информации на сети персональных компьютеров. Роль сервера в этом случае выполняет одна из рабочих станций, на котором необходимая информация открыта для многопользовательского доступа, как разделяемый ресурс.

Многие системы программирования, например, Trolltech Qt, предоставляют механизмы, позволяющие указать процедуры обработки событий, связанных с операциями над записями. Однако подобные операции требуют обеспечения работы программ без исключительных ситуаций. В случае мобильных пользователей, эпизодически включающихся в работу, гарантировать безаварийный обмен практически невозможно.

1 АЛГОРИТМ СИНХРОНИЗАЦИИ КЛИЕНТА

Для простоты рассмотрим ситуацию, когда база данных формируется на ведущей машине, а ведомая пользуется ее копией.

Обозначим S_i и C_i -- экземпляры баз данных сервера и клиента на этапе i. Здесь номер этапа соответствует некоторому моменту времени, когда рассматривается состояние всей базы данных.

Процесс использования данных можно представить кругами Эйлера, полагая, что S_i и C_i – множества уникальных кортежей. Если базы синхронизированы, то $S_i \equiv C_i$, но на этапе i+1 возникает потребность передачи сервером изменений текущей копии базы клиента. На уровне теоретико-множественных операций база данных клиента должна соответствовать

$$C_{i+1} = \left(C_i \setminus \left\{S_i \setminus S_{i+1}\right\}\right) + \left\{S_{i+1} \setminus S_i\right\}. \quad (1)$$

Однако отражающее суть этапа синхронизации выражение (1) не определяет способ его реализации. Переход $S_i \to S_{i+1}$, в общем случае, не фиксируется на логическом или физическом уровнях модели данных. Возникающая для выполнения (1) потребность наличия одновременно S_i и S_{i+1} в случае относительно незначительных изменений базы сервера, когда $\left|S_i \cap S_{i+1}\right|$ значительно больше $\left|S_i \setminus S_{i+1}\right|$ или $\left|S_{i+1} \setminus S_i\right|$, делает непривлекательным хранение сервером копии S_i до этапа $i+1$.

Далее будем рассматривать способ организации процесса синхронизации, не требующий сохранения сервером копий синхронизируемых баз данных.

Базы данных стандартно [1] допускают выполнение операций обновления, включающие добавление, коррекцию и удаление кортежей, не предполагая фиксации промежуточных состояний. В частности, изменение номера этапа $i \to i+1$ проследить нет возможности. Операции выборки данных не рассматриваются, как не меняющие состояние баз.

В отличие от формализма теории множеств, операции над базами данных могут рассматриваться более тонко. Выделим члены выражения (1):

$$D_{i+1} = S_i \setminus S_{i+1}; \quad (2)$$

$$A_{i+1} = S_{i+1} \setminus S_i. \quad (3)$$

Хотя такие члены представляют элементы множеств, процессы передачи данных клиенту различаются:

выражение (2) клиентом реализуется операцией удаления кортежей, для чего достаточно указать значение первичного ключа или номера;

выражение (3) реализуется операцией добавления кортежей, поэтому их содержимое должно быть передано клиенту.

Рассмотрим возможность синхронизации на уровне стандартных логических операций, для чего конкретизируем способ определения S_i и S_{i+1}. Потребность в этом определяется требованием логической независимости данных клиента и сервера, например, из-за использования СУБД разных типов [2].

Пусть S – экземпляр базы данных сервера. Полагаем наличие в схеме базы данных сервера первичного ключа K, в противном случае такой ключ нетрудно добавить, например, нумерацией кортежей.

Расширим схему базы данных сервера, добавляя атрибуты A и B – границы интервала активности кортежа. Значения A естественно соответствуют моменту включения кортежа в базу данных, при этом в качестве B удобно указать пустое значение или нуль. В результате набор активных кортежей соответствует условию фильтра $(A > B)$.

Потребность в расширении схемы базы данных клиента при рассматриваемых предположениях не требуется. Однако при возможности коррекции данных клиентом границы интервала активности кортежа должны быть и у клиента.

Если клиент нуждается в базе данных в момент t_{i+1}, то серверу необходимо лишь получение от клиента значения момента t_i для определения множеств

$$S_i = \{S \mid (A \le t_i) \wedge (A \le B)\} = \{S \mid (t_i \le B)\}, \quad (4)$$

$$S_{i+1} = \{S \mid (A \le t_{i+1}) \wedge (A > B)\} = \{S \mid (A > B)\}. \quad (5)$$

Выражения (4) и (5) определяют параметры (1) для операции обновления состояния базы данных любого клиента, если считать, что $C_i = \varnothing \rightarrow t_i = 0$.

Итак, можно определить шаги процесса синхронизации:

1) клиент подключается к серверу и передает момент создания копии данных

select max(A) as t from C into cursor T;

2) сервер, получив T, определяет множество подлежащих удалению кортежей

select all K from S
into table D where (t<B) and not (A>B);

3) клиент, получив D, удаляет кортежи с первичными ключами

delete all from C where K in D;

4) сервер определяет множество подлежащих добавлению кортежей

*select all * from S into table A*
where not (t<B) and (A>B);

5) клиент, получив A, выполняет добавление кортежей *insert table C from A.*

Процесс синхронизации представлен таким образом на уровне стандартных для любой СУБД средств манипулирования данными. Грубая оценка эффективности такого процесса – сокращение объема копируемых данных на отдельном шаге.

Прямолинейное копирование текущего состояния между рабочими станциями характеризуется объемом пересылаемых данных $|S_{i+1}|$, а передача данных об изменениях – $|S_i \setminus S_{i+1}| + |S_{i+1} \setminus S_i|$.

Чаще всего на практике в установившемся режиме эксплуатации систем обработки данных выполняется $|S_{i+1}| > |S_i \setminus S_{i+1}| + |S_{i+1} \setminus S_i|$.

Предлагаемая схема синхронизации удобна для одноранговых сетей, имеет возможность отката, не требует совпадения физических файлов баз данных клиентов и сервера. Существенно, что можно не требовать строгой поддержки понятий первичного ключа или уникальности кортежей. Однако логическая целостность копии базы данных у клиента критична к неделимости интервала выполнения операции синхронизации (1). Обеспечив такую неделимость, клиент может модифицировать данные своей копии и инициировать процесс синхронизации, играя роль сервера.

2 АЛГОРИТМ УПРАВЛЕНИЯ СИНХРОНИЗАЦИЕЙ

Рассмотрим далее вопрос организации управления обслуживанием множества пользователей на отдельном сервере или кэше баз данных.

Обычно в распределенной системе управления имеется возможность определения потребности в конкретных данных и назначения графика их доставки пользователю на его рабочее место согласно стратегии JIT (Just-In-Time).

Пусть в некоторый момент времени t определено множество агентов $S = \{S_i,\ i = \overline{1,m}\}$, каждая из которых характеризуется интервалом доступности во времени $[a^i, b^i]$ причем $(a^i > t) \wedge (a^i < b^i),\ i = \overline{1,m}$. Агенты S должны решить задачи из множества $\Pi = \{\Pi_j, j = \overline{1,n}\}$. Процесс решения каждой задачи может требовать использования некоторой обеспечивающей системы S_0, играющей роль элемента синхронизации всех процессов обслуживания. Интервал $[a_0, b_0]$ существования синхронизирующей системы S_0 должен перекрывать интервалы функционирования агентов, что соответствует условию

$$(a^0 < \min\{a^i,\ i = \overline{1,m}\}) \wedge (b^0 > \max\{a^i,\ i = \overline{1,m}\}).$$

Каждую из задач будем характеризовать тройкой $\Pi_j = (s_j, f_j, c_j),\ j = \overline{1,n}$, где $[s_j . f_j]$ – требуемый ин-

тервал решения задачи Π_j, а c_j – продолжительность взаимодействия агента, решающего эту задачу, с синхронизирующей системой.

Если задачи из множества Π имеют одинаковый приоритет, то порядок их решения естественно определить относительно номеров этапов взаимодействия с синхронизирующей системой S_0:

$$\Pi_i \prec \Pi_{i+1}: \quad (t_i + c_i < t_{i+1}), \quad i = \overline{1,n}.$$

Моменты времени начала решения задач, с другой стороны, связаны соотношением

$$\Pi_i \prec \Pi_{i+1}: (s_i < s_{i+1}) \vee ((s_i = s_{i+1}) \& (f_i < f_{i+1})),$$

где $i = \overline{1,n}$.

Последние условия отражают последовательный характер функционирования синхронизирующей системы S_0 и требование отсутствия прерывания отдельных фаз обслуживания

Оптимальное управление пусть соответствует минимизации смещения интервалов решения задач от идеальных значений.

Используя метод математической индукции, можно показать, что решение здесь может быть найдено следующим алгоритмом, основанном на методе динамического программирования.

Шаг 1. Фиксация начальных условий

$$\begin{aligned} &T_0 = t - 1; \\ &c_0 = 0; \\ &R_j^1 = a^j, \quad j = \overline{1,m}. \end{aligned} \tag{7}$$

Шаг 2. Прямое движение

$$\begin{cases} k^i = \arg\min\limits_j \left\{ R_j^i, \quad j \mid f_i + \max\left(0, R_j^i - s_i\right) < b^j \right\}, \\ T_i = \max\left(T_{i-1} + c_{i-1}, \quad R_{k^i}^i\right), \\ R_{k^i}^{i+1} = f_i + \max\left(0, T_i - s_i\right), \quad k^i > 0; \\ R_j^{i+1} = \max\left(R_j^i, T_i + c_i\right), \quad j \neq k^i, \quad = \overline{1,n}. \end{cases} \tag{8}$$

Шаг 3. Обратное движение

$$\begin{aligned} &t_{n+1} = b^0 + 1, \\ &t_i = \max\left(T_i, \ \min\left(t_{i+1} - c_i, \ s_i\right)\right), \quad i = \overline{n,1}. \end{aligned} \tag{9}$$

Здесь значения R_j^i представляют момент готовности агента S_j к решению задачи Π_i, $i = \overline{1,n}$, $j = \overline{1,m}$.

Упорядоченная последовательность пар $\left\{k^i, t_i\right\}$, $i = \overline{1,n}$ определяет назначение агента S_{k^i} для решения Π_i в момент времени t_i, $i = \overline{1,n}$. Если $k^i = 0$, то решение задачи Π_i в сложившихся условиях невозможно. Очевидно, что фактически обсужденное отношение вида "один ко многим" в задаче управления синхронизацией может рассматриваться как элемент представления более сложных структур.

3 СЕТЕВАЯ МОДЕЛЬ ЗАДАЧИ

Очевидно, что рассмотренная выше процедурная модель управления обслуживанием может служить основой для ее представления шаблоном функции. Однако более практичным способом является рассматриваемый далее реляционный подход к спецификации задачи в форме сети переходов. В качестве базовых моделей будем использовать расширенные сети Петри, поиск решения на которых можно проводить моделированием процесса срабатывания переходов.

Действительно, процесс функционирования системы S естественно может быть представлен в терминах временных сетей Петри *TPN* (*Timed Petri Nets*) с задержками в переходах [3]. Для представления рассматриваемой задачи в форме *TPN* (рис.1-2) введем обозначения позиций и переходов (с указанием в скобках количества их экземпляров):

$A_S(1)$ – стартовый переход; $A_F(1)$ – финишный переход; $A_1(n)$ – ожидание интервала доступности ресурса; $A_2(n)$ – интервал доступности ресурса; $A_3(n)$ – условие недоступности разделяемых ресурсов; $A_4(m)$ – подготовительные операции; $A_5(m)$ – технологические операции; $A_6(m)$ – операция синхронизации; $A_7(1)$ – интервал работы элемента синхронизации; $B_1(n)$ и $B_2(n)$ – начало и конец интервала ожидания ресурса; $B_3(n)$ – конец интервала доступности ресурса; $B_4(n)$ – разделяемый ресурс недоступен; $B_5(n)$ – разделяемый ресурс доступен; $B_6(m)$ и $B_7(m)$ – начало и конец подготовительной операции; $B_8(m)$ – простой ресурса; $B_9(m)$ – элемент синхронизации свободен; $B_{10}(1)$ и $B_{11}(1)$ – старт и финиш процесса синхронизации.

Задержки переходов рассматриваемой сети – временные параметры Π и S в относительном времени.

Поиск решения на *TPN* состоит в запуске процесса ее интерпретации, начиная со стартового перехода A_S.

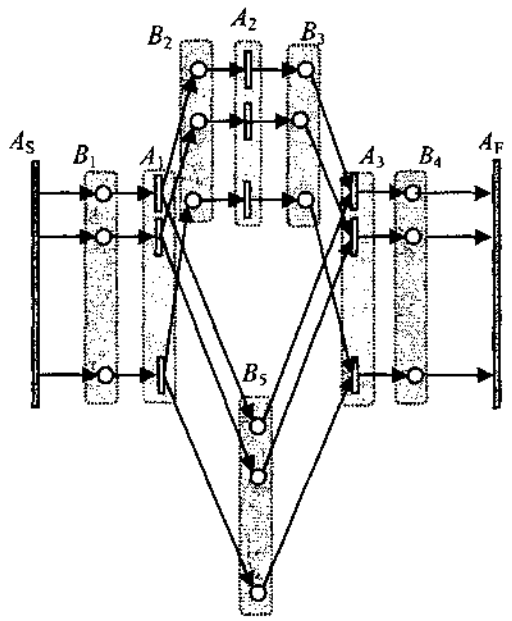

Рис. 1. Сетевая модель представления режима работы функциональных систем

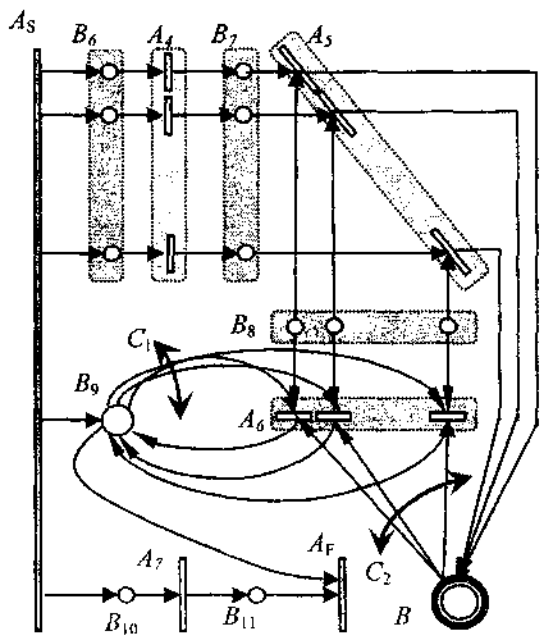

Рис. 2. Модель координации обслуживания заявок с определением закона управления назначением приоритета выходных дуг C_1 и C_2

Прямое движение в этом случае представляется рекурсивно на списке событий

$$\begin{cases} L_0 = \{(0, A_S)\}, \\ L_k = \{(f_i, a), i > k-1, a \in A\}, \ k > 0, \end{cases}$$

элементы которого – пары (f_k, a_k) однозначно определяют ассоциацию моментов времени f_k с системой, соответствующей переходу a_k [5].

Обратное движение (6) должно выполняться лишь для переходов, являющимися объектами управления – $A_5(.)$.

Поиск решения в общем случае может потребовать организации волнового процесса, но сложность его определения ниже из-за возможности локализации процесса. Функциональная спецификация задачи в форме (7)-(9) менее пригодна для детализации описания, когда требуется учесть дополнительные условия решения задач.

ЗАКЛЮЧЕНИЕ

Таким образом, процесс синхронизации кэшируемых баз данных многих мобильных рабочих мест оказывается оптимизированным по критерию минимума отклонения от предопределенных интервалов использования данных.

ЛИТЕРАТУРА

[1] Полоневич, А.М. Синхронизация кэшируемых баз данных/ М.П. Ревотюк, А.М. Полоневич//Известия Белорусской инженерной академии, № 1(17)/4, 2004. – С. 114-117.

[2] Чан, З.А. Модели оптимизации управления на основе сетей свободного выбора/М.П. Ревотюк, З.А. Чан, Е.В. Шешко //Труды V Междунар. конф. "Идентификация систем и задачи управления" SICPRO'06, Москва, 30 января – 2 февраля 2006 г.//М: Институт проблем управления им. В.А. Трапезникова РАН, 2006. – С. 2159-2170.

[3] Тихомирова, Е.В. Базовый класс интерпретатора процессов на расширенных временных сетях Петри/М.П. Ревотюк, Тихомирова Е.В.//Моделиро-вание и информационные технологии проектирования: Сб. научн. тр., вып. 4. – Мн.: ОИПИ НАН Беларуси, 2002. – С.45-56.