ПАРАЛЛЕЛИЗМ В DATALOG БАЗАХ ДАННЫХ

Дацун Н.Н.
Кафедра ПМИ, ДонГТУ
datsun@pmi.dgtu.donetsk.ua

Abstract

Datsun N.N. Concurrency in Datalog databases. *This article contains the analysis of kinds of concurrency in the Datalog- programs. Architecture of a Datalog databases machine as knowledge base machine based on logic is offered.*

Введение

Логическое программирование является как средством решения задач параллельного программирования, так и объектом применения методов параллельного программирования и аппаратной реализации. Исследуются свойства Datalog-программ с целью выявления параллелизма в механизме логического вывода и разработки архитектуры машины Datalog баз данных как машины баз знаний, основанной на логике.

1. Datalog базы данных

1.1. Дедуктивные базы данных

Под *дедуктивными базами данных (ДБД)* понимают системы, которые интегрируют технологии логического программирования и систем управления большими базами данных (БД) при создании баз знаний. Определяющим является термин *«дедуктивные»,* который отражает использование стиля логического программирования для реализации дедукции на основе содержимого БД [1].

Дедуктивную БД определяют [2] как тройку ***DBD=<EDB, IDB, IC>,*** где ***EDB*** - множество атомарных формул; ***IDB*** - множество правил вида A <- Bi, B_2... B_n, где A и B_i ($i \in [1,n]$) являются атомарными формулами, n>0; ***IC*** - множества ограничений целостности, которые могут быть любыми клаузами (утверждениями). Атомы B_1, B_2… B_n соединены между собой в теле клаузы операцией И (оператор ','). Всякий запрос к ДБД есть клауза вида <- **Qi, Q_2, ... Q_m** , где Qi ($i \in [1,m]$) может быть атомом или атомом с отрицанием.

1.2. Datalog базы данных

Дедуктивная база данных естественно определяется в терминах языка Пролог: EDB трактуется как множество истинных фактов, IDB - как множество правил, которые истинны только при выполнении входящих в них подцелей B_i, а Q_i, составляющие запрос, называются подцелями. Следующие ограничения накладываются на компоненты ДБД:

1. *EDB, IDB, IC* не должны содержать функциональных символов (ограничение для обеспечения конечности ответа на запрос пользователя);

2. Все термы, входящие в атомы клауз *EDB, IDB, IC,* могут быть переменными или константами, но не могут быть структурами (ограничение на тип клауз, которые являются *хорновскими дизъюнктами).*

ДБД с названными ограничениями называют *Datalog базами данных[2].*

1.3. Классический язык Datalog

Datalog (Дейталог) - это язык программирования ДБД (язык логического программирования с простым синтаксисом, основанным на правилах, предназначенный для взаимодействия с большими БД). В Datalog'е различают[1]: ***экстенсиональную базу данных (EDB),*** которая содержит только факты и действительно хранится на устройстве массовой памяти в отношениях реляционной базы данных (РБД) и ***интенсиональную базу данных (IDB),*** которая хранится в виде правил. ***Программой P на языке Datalog*** является множество хорновских дизъюнктов, причем любой дизъюнкт ***C*** программы ***P*** либо ***C*** є ***EDB,*** либо есть правило, удовлетворяющее следующим условиям: предикат, появляющийся в голове ***C,*** является предикатом ***IDB;*** все переменные, появляющиеся в голове С, появляются так же и в теле С. Входными данными для Datalog-программы ***P*** является ***EDB,*** а результатом - множество положительных простых основных дизъюнктов с предикатными символами ***IDB.***

2. Параллелизм в Datalog-программе

Выделим виды параллелизма в Datalog-программе и обсудим проблемы, связанные с каждым из них.

2.1. Виды параллелизма операции унификации

Механизм доказательства целевого утверждения в языках ЛП основан на *операции унификации,* которая заключается в сопоставлении аргументов «подходящих» по образцу клауз, в результате которой означиваются неконкретизированные переменные. Имеются следующие виды параллелизма операций унификации:

1. Между шаблонами исходных дизъюнктов при сопоставлении с образцом, когда некоторому целевому предикату сопоставляются несколько дизъюнков-кандидатов одновременно, у которых совпадают шаблоны ***(ИЛИ-параллелизм);***

$$\exists\, i \;=\; k\left(\forall Q_i, Q_i \in <- Q_1, Q_2, \ldots Q_m;\ \exists\, Q_k \in \left\{\vee\left(Q_k <- \wedge B_j\right)\right\}, k \in [1, n_k];\ j \in [1, p_k];\ i \in [1, m_i]\right),$$

где n_k - количество дизъюнктов с предикатным символом Q_k;

p_k - количество подцелей в k-м дизъюнкте;

m_i - количество подцелей в i-м целевом предикате.

2. Между целевыми предикатами - если в цели имеется несколько подцелей, то операция унификации для каждого целевого предиката может выполняться параллельно ***(И-параллелизм);***

$$\forall Q_i . Q_i \in <- Q_1, Q_2, \ldots Q_m\ \exists Q_k\ \in \{\, Q_k <- \wedge B_j\}, k \in [1, n_k]; j \in [1, p_k];\ i \in [1, m_i]$$

3. Между аргументами двух клауз, сопоставляемых параллельно ***(параллелизм собственно операции унификации).***

$$\forall Q_i . Q_i \in <- Q_1\ (t_1\ , t_2, \ldots t_{r1},), Q_2\ (t_1\ , t_2, \ldots t_{r2},), \ldots\ Q_m\ (t_1\ , t_2, \ldots t_{rm},),$$

$$\exists \mathrm{t_i} = \mathrm{g}_i \exists Q_k \in \{\mathrm{V}(Q_k(g_1, g_2 \ldots g_{rk}), < -\wedge B_j)\}, \mathrm{k} \in [1, n_k];\ \mathrm{j} \in [1, p_k];\ \mathrm{i} \in [1, m_i];\ \mathrm{l} \in [1, r_l]$$

Представление Datalog-программы в виде И-ИЛИ-дерева наглядно иллюстрирует наличие в ней названных видов параллелизма. Ставя в соответствие каждому узлу И-ИЛИ-дерева один процесс, можно обеспечить параллельную обработку. Введем следующие термины. Процессы в узлах И, соответствующие головам хорновских дизъюнктов, назовем *исходными процессами.* Процессы, запущенные с И-параллелизмом в И-ветвях, назовем *порожденными процессами.*

2.2. И - параллелизм

Предикаты, стоящие в вершинах ветвей И, логически осуществляются параллельно. Исходные процессы поддерживаются до тех пор, пока не завершатся успешно все процессы, запущенные с И-параллелизмом. При завершении хотя бы одного порожденного процесса неуспехом, результат осуществления исходного процесса также считается безуспешным, и этот результат может быть возвращен в узел, расположенный выше. Здесь существуют следующие проблемы:

1.Необходимость классического *механизма синхронизации* между исходным и порожденными им процессами.

2.Необходимость механизма, который сообщал бы о неудаче осуществления конкретного порожденного процесса всем другим порожденным процессам (которые пока активны) и прекращал бы их выполнение. Это метод повышения эффективности работы механизма доказательства целевого утверждения. В отличие от Пролога, который чувствителен к порядку следования подцелей в целевой клаузе, И-параллелизм Datalog'a предусмотрен на уровне вычислительных моделей языка.

3.Разрешение ситуации, когда получены разные результаты осуществления порожденных процессов при наличии у них *общей переменной* у различных подцелей целевого предиката (так называемая *резолюция конфликта).*

2.3. Струйный параллелизм

В общем случае проблема резолюции конфликта трудноразрешима. Одним из вариантов решения этой проблемы является следующий подход [5]. Сначала исследуются свойства переменной, принадлежащей порожденным процессам. На основе этого анализа производят разделение порожденных процессов на группу процессов, которые должны выполняться последовательно, и группу процессов, которые могут быть выполнены параллельно. В соответствии с такой классификацией пытаются осуществить порожденные процессы. Если использовать параллелизм выполнения ветвей Datalog-программы, то группу порожденных процессов, обычно выполняемых последовательно, окажется возможным выполнить параллельно. Это третий вид параллелизма, называемый *струйным параллелизмом.*

2.4. ИЛИ - параллелизм

Ветви ИЛИ отражают недетерминированность выполнения программы на Datalog’е, т.е. логически может быть выбрана любая ветвь. Здесь существуют следующие проблемы:

1. Необходимость копирования и передачи условия выполнения (прежде всего, информации для унификации) из исходного узла в порожденные узлы, находящиеся в

вершинах соответствующих ветвей, причем проблема заключается в повышении стоимости копирования по мере увеличения объема передаваемой информации;

2 Необходимость обеспечения синхронности между исходным процессом и группой порожденных процессов (после получения одного решения от порожденного процесса оно пересылается исходному процессу; после нового требования о передаче, поступившего от исходного процесса, посылается следующее решение и т.д.).

Кроме этого, в Datalog-программе ИЛИ-параллелизм может быть использован при нахождении множества дизъюнктов-решений (в отличие от Пролога [4], где количество находимых решений чаще всего определяется программистом, при вычислении Datalog-программы **находятся все дизъюнкты решения**) [1]

2.5. Параллелизм собственно операции унификации

Предикат клаузы обычно имеет несколько аргументов. Операции унификации набора аргументов, соответствующих предикату, могут осуществляться параллельно. Здесь существуют следующие проблемы, которые иллюстрирует пример 1, представляющий ***EDB*** и Datalog-программу ***P*** над ней:

Пример 1:

EDB: **{ q(23, 60, 0). q(0,1, 5).}**

P: **{p(0, X, X):- q(0, 1, X). p(S, Y, Z):- q(23,60, S). }**

1. Случай, когда несколько аргументов, принадлежащих одному предикату, имеют общую переменную (первое правило программы ***P***, переменная ***X);***

2. Случай одноименных переменных в голове и теле клаузы для различных аргументов, принадлежащих одному предикату (второе правило программы ***P).***

В этих случаях, так же как и в случае И-параллелизма, необходимо предусматривать меры для разрешения конфликтов.

Таким образом, язык Datalog обладает четко выраженной структурой, обеспечивающей четыре вида параллелизма. Параллелизм Datalog-программы можно определить, проанализировав свойства этой программы. Поэтому в состав архитектуры машины для Datalog баз данных предлагается ввести анализирующий препроцессор для распараллеливания Datalog -программ.

3. Об аппаратной реализации параллелизма в Datalog базах данных

Аппаратная реализация Datalog баз данных представляют собой классическую машину баз знаний [6] с компонентами:

1. Спецпроцессор логического вывода (СЛВ);

2. Машина баз данных (МБД).

Вопросы аппаратной поддержки параллелизма в каждой из названных компонент обсуждаются в работах [3, 6, 7, 8]. Важнейшим остается вопрос связывания СЛВ и МБД (или CPR - Coupling Prolog to Relational databases) Различают два варианта связывания Пролога (как логического вывода) и РБД [1]: слабое (Loosely CPR-системы) и сильное (Tightly CPR-системы), причем речь идет о передаче механизму логического вывода из РБД либо одного кортежа, либо всех кортежей, удовлетворяющих запросу.

В данной работе в Datalog базах данных в качестве интерфейса между СЛВ и МБД предлагается использовать реляционную алгебру (RA).

3.1. Реляционная алгебра - интерфейс между СЛВ и МБД в машине Datalog баз данных

С одной стороны, операции RA получили наиболее эффективную реализацию на параллельных архитектурах. Аппаратная реализация операций RA представлена следующими вариантами параллельных архитектур:

1. Реляционный процессор БД с конвейерной потоковой обработкой, сочетающей достоинства горизонтального параллелизма при выполнении одной операции с функциональным параллелизмом при выполнении последовательности операций и транзакций[3];

2. Сопроцессор потоковой сортировки отношений для слияния отношений или компараторные сопроцессоры[3];

3. Потоковая фильтрация данных в множестве каналов устройств массовой памяти [3, 8];

4. Многоуровневая многопроцессорная МБД с высокоэффективными системами коммутации [3];

5. Большие РБД на транспьютерах [9].

С другой стороны, любой Datalog-программе может быть поставлена в соответствие система уравнений реляционной алгебры. Выразительная мощность классического Datalog'a эквивалентна выразительности положительной реляционной алгебры. Любая Datalog-программа транслируется в систему уравнений RA [1].

Поэтому в данной работе в состав машины для аппаратной реализации Datalog баз данных предлагается ввести конвертор Datalog-RA, выполняющий роль интерфейса между СЛВ и МБД.

3.2. Архитектура машины Datalog баз данных

Предлагается следующая архитектура машины Datalog баз данных, использующая параллелизм операций унификации СЛВ и операций реляционной алгебры в МБД.

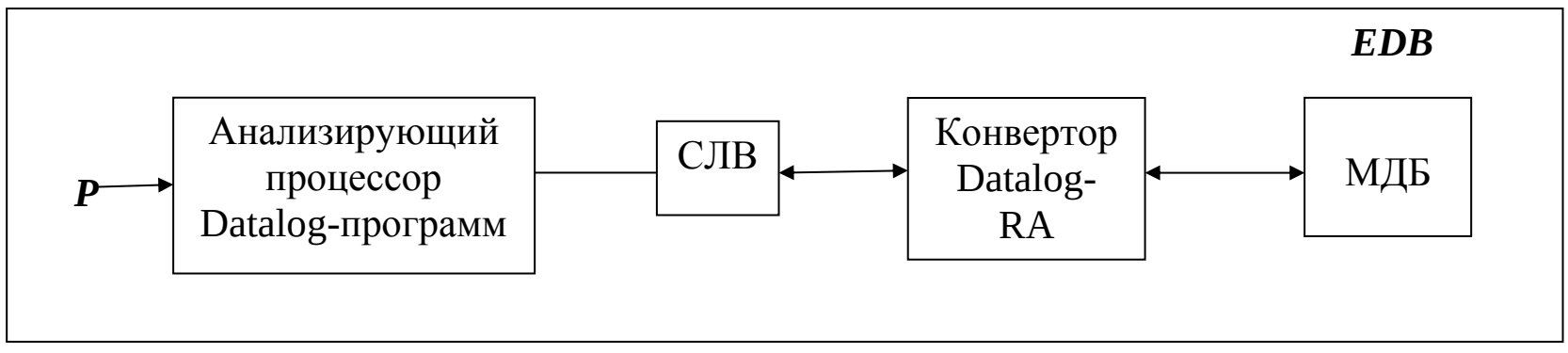

Рисунок 3.1 - Архитектура машины Datalog баз данных

Заключение

Повышение эффективности баз знаний, основанных на логическом подходе, может быть обеспечено аппаратной поддержкой их основных компонент и аппаратной

реализацией функций этих компонент. На основе исследования параллелизма в ДБД предложена архитектура машины Datalog баз данных.

Литература

1. Ceri S., Gottlob G., Tanca L. Datalog: a self-contained tutorial// Программирование. - 1991, N 5. - с.9-31.
2. Фернандес Х.А., Минхер Д. Теория и алгоритмы дизъюнктивных дедуктивных баз данных// Программирование. - 1993, N1. - с.5-39.
3. Искусственный интеллект: Кн.3. Программные и аппаратные средства: / Под ред. В.Н. Захарова, В.Ф. Хорошевского. - М.: Радио и связь, 1990. - 368с.
4. Клоксин Н., Меллиш К. Программирование на языке Пролог. - М.: Мир, 1987. - 336с.
5. Компьютеры на СБИС: Кн. 2/ Мотоока Т., Хорикоси Х., Сакаути М. и др. - М.: Мир, 1988. - 336 с.
6. Калиниченко Л.А., Рывкин В.М. Машины баз данных и знаний. - М.: Наука, 1990. - 296с.
7. Анамия М., Танака Ю. Архитектура ЭВМ и искусственный интеллект. - М.: Мир, 1993. - 400с.
8. Озкарахан Э. Машины баз данных и управление базами данных. - М.: Мир, 1989. - 696с.
9. Жуков И.А. Аппаратная реализация систем больших баз данных на основе транспьютеров// Управляющие системы и машины. - 1997, N1/3. - с. 100-107.