УДК 004.942+004.272.26

МОДЕЛИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ТРАНСЛЯТОРЕ ЯЗЫКА ЛОГИЧЕСКОГО ПРОГРАММИРОВАНИЯ

Н.Н. Дацун, А.В. Шуликов
Донецкий национальный технический университет

С помощью транслятора языка логического программирования выполнено моделирование ИЛИ-параллелизма с полной прозрачностью на вычислительном кластере из 4 ядер. Получены прогнозные значения времени выполнения при увеличении количества процессорных ядер.

1. Параллелизм в логической программе и подходы к реализации

В параллельных трансляторах логического языка выделяют следующие виды параллелизма:

а) «И»-параллелизм (подцели, принадлежащие данному запросу выполняются параллельно различными вычислительными модулями);

б) «ИЛИ»-параллелизм (одна подцель может быть унифицирована с заглавиями нескольких клауз, и тела этих клауз вычисляются различными вычислительными модулями);

в) мультитрединг.

Процесс распараллеливания логической программы может быть построен по одному из трех принципов:

а) полная прозрачность - процесс распараллеливания происходит внутри транслятора за счет особой организации внутреннего представления данных и процесса получения результата, т.е. происходит без участия программиста;

б) промежуточная прозрачность - программист должен каким–то образом декларировать предикаты распараллеливания, после чего можно работать с ними, как и с любыми другими предикатами. В этом случае остается скрытым факт, что во время сопоставления с предикатом используется механизм распараллеливания;

в) отсутствие прозрачности - в программе на логическом языке должны быть явно декларированы операторы языка параллельных вычислений.

Реализация любого вида параллелизма и использование любого из принципов прозрачности требуют модификации всех стандартных блоков транслятора логического языка.

В современных трансляторах языков ЛП можно выделить реализации И-параллелизма - Parlog Parallel [1]; ИЛИ-параллелизма -

Акторный Пролог [2], YAPOr [3]; мультитрединга – QuProlog [4], SWI Prolog [5], Arity Prolog [6], BinProlog [7]. Транслятор Brain Aid Prolog [8] предназначен для транспьютерных сетей. В Parlog, QuProlog, Акторном Прологе, SWI Prolog реализована промежуточная прозрачность, а в YAPOr – полная прозрачность параллелизма.

Параллелизм логических программ может быть использован одинаково как SIMD-архитектурах, так и на MIMD- архитектурах. На SIMD-архитектурах более естественным является ИЛИ-параллелизм. Авторами разработан транслятор языка логического программирования с полной прозрачностью ИЛИ-параллелизма. С его помощью выполнено моделирование характеристик вычислительного кластера для решения задач определенного вида.

2. Описание моделирующего стенда

Для проведения моделирования использовался кластер из двух компьютеров. Суммарно было доступно четыре процессорных ядра, что позволило оценить увеличение производительности с увеличением количества процессорных ядер и сделать прогноз на последующее расширение кластера.

2.1 Аппаратное обеспечение моделирующего стенда.

Тесты проводились на 2 компьютерах, взаимодействующих по локальной сети стандарта FastEthernet. Первый компьютер имел процессор Intel Core2Duo E6550 с тактовой частотой 2.33 ГГц и двумя ядрами, объем оперативной памяти 2 Гб. Второй компьютер имел процессор Intel Core2Duo E7200 с тактовой частотой 2.53 ГГц и двумя ядрами, объем оперативной памяти 2 Гб.

2.2 Программное обеспечение моделирующего стенда

На обоих тестовых компьютерах была установлена операционная система OpenSUSE Linux версии 11 с ядром Linux версии 2.6.25.16. Для организации взаимодействия процессов использован MPICH2 версии 1.0.7.

2.3 Организация процесса моделирования

Для определения среднего значения времени на определенном наборе данных проводилось 50 запусков для каждого набора и вычисление среднего значения. В ходе эксперимента учитывалось только время от получения внутреннего представления запроса пользователя до момента получения результатов, без учета времени

вывода результатов пользователя. Таким образом, учитывалось только время на проведение унификации (в параллельном варианте с учетом времени формирования и рассылки задач, а также сбора результатов).

2.3.1 Вид задачи и генерация тестовых наборов

Для проведения экспериментов использовалась автоматическая генерация тестовых программ. При помощи генерирующего скрипта на языке программирования Perl генерировалась экстенсиональная база данных (ЭБД) с предикатом “pred” и тремя псевдослучайными положительными целочисленными аргументами. Факты с одинаковым набором аргументов не допускались. Генерировались ЭБД с количеством от 2000 до 5000 фактов с шагом 500. Затем в тест добавлялась интенсиональная база данных (ИБД) из однотипных правил с тремя аргументами, содержащих в теле правила три предиката с аргументами. В каждой тестовой программе в ИБД содержалось 160 альтернатив правила.

2.3.2. Автоматизация процесса моделирования

Автоматизация процесса выполнения тестов проводилась за счет модификации кода программы:

а) интерактивный режим был заменен на единоразовое получение запроса и выполнение в цикле 50 повторных унификаций с выводом результатов времени для каждого теста и среднего значения по всем шагам;

б) тест выполнялся при помощи скрипта на языке Perl;

в) запрос к программе вида “rule(X,Y,Z)” перенаправлялся из текстового файла;

г) вывод результатов тестов производился в текстовые файлы. К имени файла с программой добавлялось количество ядер, на которых выполнялся тест.

3. Результаты моделирования

3.1 Время выполнения в зависимости от количества фактов и количества ядер

Тестирование проводилось для ЭБД в составе 2000, 2500, 3000, 3500, 4000, 4500, 5000 фактов и ИБД из 160 правил. Количество ядер менялось от 1 до 4. Обобщенные результаты в виде средних значений в зависимости от количества фактов (KF) и ядер (KY) приведены в таблице 3.1. В ней приведены уравнения линий тренда и достоверности для всех тестовых наборов данных, а также

прогнозные значения количества ядер, полученные при помощи уравнений линий трендов.

Таблица 3.1

Зависимость времени выполнения от количества фактов и ядер, уравнения линии тренда и прогнозное количество ядер

Коли-чество фактов (KF)	Время выполнения (сек)							Уравнения линий тренда	Досто вер-ность (R^2)
	Количество ядер при моделировании (KY)				Прогнозное количество ядер				
	1	2	3	4	5	6	7		
2000	11,16	5,87	4,18	3,25	2,64	2,25	1,96	$y = 11.06x^{-0.8894}$	0.9995
3000	25,67	13,15	9,07	6,92	5,59	4,70	4,07	$y = 25.556x^{-0.9445}$	0.9999
4000	48,05	24,39	16,82	12,65	10,23	8,59	7,41	$y = 47.889x^{-0.9593}$	0.9998
5000	79,68	40,42	28,32	21,29	17,25	14,52	12,55	$y = 79.226x^{-0.9471}$	0.9995

Результаты моделирования показывают, что с дальнейшим увеличением количества ядер время выполнения будет уменьшаться пропорционально количеству ядер. Так в случае ЭБД при KF=2000 фактов на 7 ядрах время отклика составляет менее 2 сек. При этом с увеличением количества ядер с 6 до 7 и с 5 до 6 прирост производительности не превышал 0.5 сек. Для большинства задач этого вполне достаточно при интерактивном получении данных из базы данных логической программы. Значит, для рассматриваемой ЭБД целесообразно KY=5.

3.2 Время выполнения в зависимости от количества правил и количества ядер

При оценке времени выполнения унификации при увеличении количества правил следует учитывать, что ситуация отличается от варьирования количества фактов лишь тем, что в случае с фактами возрастает время вычисления одной альтернативы, а в случае правил – возрастает количество альтернатив. За основу взята тестовая программа, содержащая ЭБД из 2500 фактов. Количество альтернатив правила в ИБД программы варьировалось от 320 до 800 с шагом 160. Количество ядер менялось от 1 до 4. Обобщенные результаты в виде средних значений в зависимости от количества правил (KA) и ядер (KY) приведены в таблице 3.2.

Результаты моделирования показывают существенное возрастание времени унификации с увеличением количества альтернатив. Для случая последовательной реализации транслятора (1 ядро) получено уравнение линии тренда $y = 34.265x^{0.6492}$ (R^2=0.9954). С его помощью получены экспериментальные значения

времени выполнения унификации на одном ядре в зависимости от количества альтернатив с прогнозом на несколько шагов вперед.

Таблица 3.2

Зависимость времени выполнения от количества правил и ядер

Количество альтернатив (КА)	Время выполнения (сек)			
	Количество ядер (КУ)			
	1	2	3	4
320	34,91	17,76	12,13	9,26
480	52,01	26,35	18,02	13,81
640	69,20	35,02	24,35	18,53
800	86,38	44,04	30,31	23,52

Также результаты моделирования показывают, что распараллеливание процесса унификации различных альтернатив правил ИБД ведет к значительному увеличению скорости работы логической программы.

Выводы

С помощью транслятора языка логического программирования выполнено моделирование ИЛИ-параллелизма с полной прозрачностью на вычислительном кластере из 4 ядер. Получены характеристики времени выполнения программ в зависимости от размера ЭБД, ИБД и количества ядер. Получены прогнозные значения времени выполнения при увеличении количества процессорных ядер. Предложенный подход к моделированию ИЛИ-параллелизма может использоваться при расчете характеристик кластера для решения задач распараллеливания выполнения логической программы.

Литература

1. Gregory S. Parallel Logic Programming in PARLOG, The Language and Its Implementation. - Boston: Addison-Wesley Longman Publishing, 1987.
2. Акторный Пролог. Определение языка программирования. - Режим доступа: http://www.cplire.ru/Lab144/koi8/09010000.html
3. Yet Another Prolog. - Режим доступа: http://www.dcc.fc.up.pt/~vsc/Yap/
4. Qu-Prolog Home Page. - Режим доступа: http://www.itee.uq.edu.au/~pjr/HomePages/QuPrologHome.html
5. SWI Prolog Home Page. - Режим доступа: http://www.swi-prolog.org/
6. Arity Prolog Home Page. – Режим доступа: http://www.arity.com/
7. BinProlog Home Page. – Режим доступа: http://www.binnetcorp.com/BinProlog
8. Brain Aid Prolog Home Page. – Режим доступа: http://www.comnets.rwth-aachen.de/~ost/private.html

Получено 28.05.09