ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ ЭКСТЕНСИОНАЛЬНОЙ ЧАСТИ ДЕДУКТИВНЫХ БАЗ ДАННЫХ

Пушкаренко С.А., Дацун Н.Н.
Кафедра прикладной математики и информатики ДонНТУ
S.Pushkarenko@it.stirol.net

Дедуктивные базы данных (ДБД) хранят не только данные, но и знания. Машина логического вывода позволяет обрабатывать знания, представленные в различных моделях: семантические сети, фреймы, продукционные правила и т.д.

В ходе работы были проведены исследования способов распараллеливания запросов к экстенсиональной части ДБД, представленной в виде фактов логической программы.

Для исследования способов распараллеливания и оценки их эффективности были выбраны две различных парадигмы параллелизма [1].

В режиме параллельного выполнения программы, можно вывести две формулы затраты времени, для координирующего процессора с рангом 0 и остальных процессоров. Формула для ведущего процессора будет выглядеть следующим образом:

$$t_{main}=t_{посл.ч.}+t_{форм.п.}+t_{отпр.п.}+t_{уд.вып(max)}+t_{выд.рез.} \quad (1)$$

где $t_{посл.ч.}$ – время выполнения процессором последовательной части,

$t_{форм.п.}$ – время формирования пакетов информации для рассылки другим процессорам,

$t_{отпр.п.}$ – время отправки сообщений и общих данных каждому процессору,

$t_{уд.вып.(max)}$ – время удалённого выполнения и сбора результатов,

$t_{выд.рез.}$ – время на анализ и выдачу окончательного результата пользователю.

Время выполнения процессором последовательной части зависит от производительности ведущего процессора и размеров задания. Время формирования пакетов информации для рассылки зависит от оптимальности выбранных структур данных в трансляторе. Время отправки сообщений и общих данных процессорам зависит от количества данных и скорости работы сети и организации кластера или многопроцессорной системы. Время удалённого выполнения и сбора результатов можно представить формулой:

$$t_{уд.вып.(max)} = max\ (t_{ожид.п.[i]}+t_{обр.п.[i]}+t_{лок.обр.[i]}+t_{отс.рез.[i]})\ i=1..n-1 \quad (2)$$

где $t_{ожид.п.[i]}$ – время ожидания i-ым процессором данных,

$t_{обр.п.[i]}$ – время обработки полученных сообщений i-ым процессором,

$t_{лок.обр.[i]}$ – время локальной обработки данных и операций,

$t_{отс.рез.[i]}$ – время формирования пакета с результатами работы и отсылки его i-ым процессором ведущему узлу кластера или процессору,

n – общее количество процессоров, используемое для данной задачи.

Максимальное время $t_{уд.вып.(max)}$ будет зависеть от производительности наиболее медленного узла и загруженности сети, а время формирования пакета с результатами также будет зависеть от типа запроса, наиболее быстрыми будет запрос по константе, формирующий ответ «да» или «нет», наиболее медленным – запрос по результату с большим количеством кортежей.

Для оценки эффективности работы параллельной системы удобнее всего использовать формулы 3 и 4:

$$S = T(l)/T(p) \quad (3)$$

где S – ускорение,

$T(l)$ – время выполнения в последовательном режиме,

$T(p)$ – время выполнения в параллельном режиме.

$$E(p) = S / p \quad (4)$$

где $E(p)$ – эффективность параллельного алгоритма на p процессоров,

p – количество процессоров в системе [2].

Для оценки эффективности были проведены замеры времени выполнения программы в зависимости от количества фактов в программе и количества аргументов в предикате факта.

В зависимости от количества фактов по данному предикату запроса с постоянным количеством аргументов (N=3) были получены следующие результаты:

Количество фактов	Лексический анализ	Синтаксичес-кий анализ	Внутреннее представление	Унификация	Всего
10	0.00373763	0.00632175	0.00147030	0.00115070	0.0126804
15	0.00319286	0.00607507	0.00163624	0.00114903	0.0120532
20	0.00319678	0.00609295	0.00148566	0.00114065	0.0119160
25	0.00321450	0.00611340	0.00156231	0.00115840	0.0120486
30	0.00324013	0.00614560	0.00162104	0.00117213	0.0121789
35	0.00328123	0.00618091	0.00169100	0.00120112	0.0123543
40	0.00340102	0.00627505	0.00178221	0.00128212	0.0127404

В зависимости от количества аргументов предиката и соответственно количества аргументов запроса (K) при постоянном количестве фактов (N=40), были получены результаты:

Количество аргументов	Лексический анализ	Синтаксический анализ	Внутреннее представление	Унификация	Всего
1	0,00234137	0,00427893	0,00101024	0,00092315	0,0085547
2	0,00284628	0,00583287	0,00128764	0,00113432	0,0111011
3	0,00319873	0,00644551	0,00149377	0,00131357	0,0124516
4	0,00398731	0,00811222	0,00192123	0,00152312	0,0155439
5	0,00812201	0,01521874	0,00362584	0,00250456	0,0294711

Время последовательного выполнения программы существенно зависит от количества аргументов предиката, использующегося в запросе, и увеличивается экспоненциально, по формуле $y=0.006e^{0.281}$ с достоверностью в 0.902.

Для моделирования параллелизма с помощью MPICH, была выбрана модель гомогенного симметричного кластера (HSMP). Использовались следующие команды библиотеки MPI: MPI_Init, MPI_Comm_size, MPI_Comm_rank, MPI_Get_processor_name, MPI_Initialized, MPI_Finalize, MPI_BCast, MPI_Gather. Для определения времени выполнения различных фаз трансляции была использована комманда MPI_WTime.

Для распараллеливания в процессе трансляции была выбрана фаза унификации. Распараллеливание каждого запроса было выполнено по количеству его аргументов. Общее количество аргументов делится на количество процессоров, и, таким образом, распределяется объём работы, выполняемый каждым процессором параллельной системы. Формируются необходимые пакеты информации, общих данных, и с помощью команды MPI_Bcast происходит рассылка пакетов в соответствии с рангом назначенного для их выполнения процессора. Если количество процессоров системы больше, чем количество аргументов запроса, выводится сообщение о неоптимальности использования ресурсов кластера. Процессоры системы выполняют назначенные им части этапа унификации и возвращают результаты этого процесса. Координирующий процессор собирает результаты выполнения удалённой части с помощью команды MPI_Gather, формирует результат унификации и выводит его на экран. Затем собирает коды ошибок, найденных в процессе трансляции и также выводит их на экран с расшифровкой их значений. Далее ведущий процессор переходит к унификации следующего запроса или заканчивает действие программы, освобождая ресурсы параллельной системы. Оптимальность распараллеливания была определена при помощи моделирования гомогенного кластера, состоящего из 3-х одинаковых процессоров. Конфигурация, используемых элементов кластера:

процессор Intel Centrino 2.0 Ghz;
жёсткий диск SATA 5400rpm, 2mb cache;
1Gb оперативной памяти DDR2;
100Mbit сетевая карта.

Для замеров производительности была использована программа с постоянным количеством фактов (N = 40) и переменным количеством аргументов (K) запроса по этим фактам. Были получены следующие результаты производительности системы:

K	Л А	С А	В П	Унификация на первом процессоре	Унификация на втором процессоре	Унификация на третьем процессоре	Всего
1	0,00234137	0,00427893	0,00101024	0,00081301	0,00232871	-	0,0107723
2	0,00284628	0,00583287	0,00128764	0,00098574	0,00232781	0,00228379	0,0122803
3	0,00319873	0,00644551	0,00149377	0,00103478	0,00318782	0,00222149	0,0153606
4	0,00398731	0,00811222	0,00192123	0,00113782	0,00321298	0,00336156	0,0185201
5	0,00812201	0,01521874	0,00362584	0,00138726	0,00409271	0,00342112	0,0324466

Ниже приведена сравнительная таблица результатов выполнения программы на однопроцессорной системе и параллельной системе из трех процессоров.

Таблица 1 - Сравнительная таблица результатов выполнения программы

Кол-во аргументов (K)	1 процессор	3 процессора
1	0,0085547	0,0107723
2	0,0111011	0,0122803
3	0,0124516	0,0153606
4	0,0155439	0,0185201
5	0,0294711	0,0324466

Полученные результаты показывают, что данная архитектура и данный способ распараллеливания в процессе трансляции будет более оптимален и даст прирост производительности при большем количестве аргументов в предикате запроса, что характерно для более сложных запросов и отношений между фактами программы. Время выполнения программы на трёх процессорах будет зависеть от количества аргументов биномиально по формуле $y=0.001x^2-0.002x + 0.015$ с достоверностью в 0.641. Невысокая достоверность аппроксимации вызвана небольшим количеством аргументов, по которым проводились замеры. Регрессия на этом промежутке особенно высока, она уменьшается с ростом количества аргументов.

Параллельное выполнение на трех процессорах начинает давать прирост производительности при количестве аргументов запроса более десяти.

В результате проведённых исследований были сделаны следующие выводы:

- задержки передачи пакетов в сети достаточно велики;
- поддерживаемые форматы данных в MPI и количество общих ресурсов памяти на этапе унификации не способствуют увеличению производительности программы в параллельном режиме;
- большие затраты времени наиболее характерны для этапа синтаксического анализа.

На основании вышеизложенного можно заключить, что наиболее эффективным является распараллеливание логической программы на этапе синтаксического анализа. Это позволит решить следующие проблемы:

- объем передаваемых удалённым процессорам данных будет сведен к минимуму, потому скорость формирования и передачи пакетов будет намного выше;
- распараллеливание можно выполнять не по количеству аргументов программы, а по фактам программы, что увеличит степень производительности и степень распараллеливания этапов трансляции, но в таком случае понадобится дополнительная синхронизация на этапе внутреннего представления;
- эффективность распараллеливания будет тем выше, чем меньше задержка при передаче пакетов;
- степень распараллеливания будет высока даже при небольшом размере программы и большом количестве интерпретирующих её процессоров, что обеспечит возможность выполнения задач с малым количеством фактов (N) и аргументов (K), требующих выдачи результатов в реальном времени с наименьшей задержкой.

Литература

1. Jost G., Jin H., Mey D., Hatay F. Comparing the OpenMP, MPI, and Hybrid Programming Paradigm on an SMP Cluster, 2004.

2. Головкин Б. Вычислительные системы с большим числом процессоров.-М.: Радио и связь,1995.