Моделирование операции соединения, выполняемой на мультипроцессоре баз данных, оснащенном многоядерными сопроцессорами*

П.С. Костенецкий

Южно-Уральский государственный университет

В работе предлагается математическая модель мультипроцессора баз данных, оснащенного многоядерными сопроцессорами, позволяющая моделировать передачу данных между сопроцессором и основной памятью в сжатом виде. Модель включает в себя модель аппаратной платформы, модель операционной среды и стоимостную модель. На базе предложенной модели разработан эмулятор, позволяющий оценить применимость методов сжатия при передаче данных между сопроцессором и основной памятью.

В настоящее время одним из перспективных направлений развития параллельных систем баз данных [12] являются использование гибридных вычислительных комплексов с узлами, содержащими многоядерные сопроцессоры [1]. Для исследования произвольных многопроцессорных конфигураций обычно используются математические модели [8, 9]. В работе предложена новая математическая модель мультипроцессора баз данных, оснащенного многоядерными сопроцессорами, являющаяся расширением модели *DMM* [7]. Модель включает в себя модель аппаратной платформы, модель операционной среды и стоимостную модель.

Модель аппаратной платформы. Основными элементами модели аппаратной платформы являются процессорный модуль, сопроцессорный модуль и модуль коммутатора. В рамках данной модели, аппаратные архитектуры систем баз данных [2] представляются в виде деревьев, далее называемых *HDM*-деревьями. Узлы *HDM*-деревьев представляют собой процессорные модули, сопроцессорные модули и модули сетевого коммутатора. Ребра *HDM*-дерева соответствуют каналам передачи данных. На HDM-дерево накладываются следующие ограничения:

- корнем HDM-дерева может быть только модуль сетевого коммутатора;
- листьями HDM-дерева могут быть только сопроцессорные модули;
- сопроцессорные модули могут быть связаны с сетевым коммутатором только посредством процессорного модуля.

На рис. 1 представлен пример моделируемой аппаратной конфигурации системы баз данных. Система представляет из себя кластер с вычислительными узлами, содержащими многоядерные ускорители.

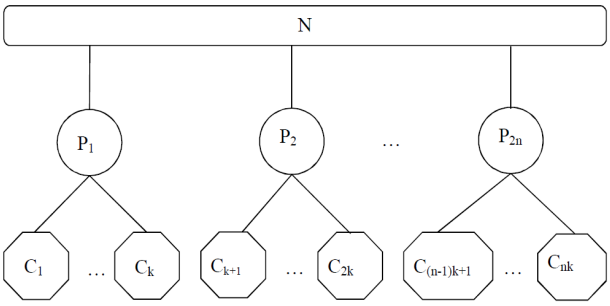

Рис. 1. Пример аппаратной конфигурации в виде *HDM*-дерева

Здесь *N* – модуль коммутатора, *P* – процессорные модули, *C* – сопроцессорные модули, ребра дерева соответствуют каналам передачи данных.

* Работа выполнена при финансовой поддержке Минобрнауки РФ в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014—2020 годы» (Соглашение № 14.574.21.0035).

Предполагается, что к моменту начала выполнения запроса база данных уже целиком равномерно распределена в оперативной памяти кластера между всеми процессорными модулями. Обработка операции соединения производится только на сопроцессорных модулях.

Модель операционной среды. Минимальной единицей данных в предлагаемой модели будем считать кортеж. Допустим все кортежи имеют одинаковый размер. При обработке кортежи могут делиться сопроцессорным модулем на две группы – *свои* и *чужие*. *Свои кортежи* – это кортежи, которые должны быть обработаны на данном сопроцессорном модуле. *Чужие кортежи* должны быть обработаны на других сопроцессорных модулях. Номер сопроцессорного модуля, которому необходимо передать чужие кортежи определяется с помощью функции пересылки.

Время передачи данных между вычислительным узлом и сопроцессором можно уменьшить, применив сжатие данных в СУБД [3, 4, 6, 11]. Будем считать, что данные в моделируемой СУБД хранятся в сжатом виде. Данные разбиты на фрагменты, называемые *томами*. Размер тома T зависит от количества кортежей, содержащихся в томе, и коэффициента сжатия данных. Коэффициентом сжатия данных будем называть отношение объема сжатых данных к объему исходных данных. Коэффициент перекоса ω указывает долю *своих* кортежей в томе. Время работы модели разбивается на *такты.*

Пусть один кортеж сжимается за время q. Тогда время создания тома данных, содержащего n кортежей будет равно: $t_c = nq$. Время распаковки данных $t_d = t_c * v$, где v – константа, зависящая от выбранного алгоритма сжатия.

Стоимостная модель. Для вычисления времени работы моделируемой системы баз данных необходимо учитывать скорость передачи данных. Пусть s_n – скорость передачи данных на n-ом уровне дерева. Тогда время передачи одного тома между двумя сопроцессорными модулями, подключенными к различным процессорным модулям, будет: $S = 2(s_1 + s_2)$. Таким образом, длительность одного такта будет равна:

$$t_i = s_2 + t_d + t_f + t_c p + S + (1-p)t_d + t_{\bowtie} + t_c + s_2$$
$$= 2s_1 + 4s_2 + (2v - vp + p + 1)n\omega + t_f + t_{\bowtie};$$

где t_c – время упаковки тома, t_d – время распаковки тома, ω – коэффициент перекоса, t_f – время выполнения функции фрагментации, $t_{\bowtie}$ – время выполнения операции соединения, n – количество кортежей, содержащихся в томе, v – константа, отражающая зависимость между временем сжатия и распаковки.

Время работы мультипроцессора баз данных с многоядерными ускорителями при обработке одной транзакции равно сумме времени, потраченного на все такты работы модели:

$$T = \sum_{i=1}^{M} t_i.$$

Моделирование операции соединения. Рассмотрим процесс моделирования операции соединения в модели на примере алгоритма естественного соединения с хешированием *Main memory Hash Join (MHJ)*. На рис. 2. изображен псевдокод последовательного алгоритма операции соединения с хешированием.

```
for each r in R
    вычислить значение хеш-функции для r по атрибуту A
    добавить r в хеш-таблицу
end for
for each s in S
    вычислить значение хеш-функции для s по атрибуту A
    for each r in хеш-таблица
        if join(r, s)   R S
            return (r, s)
    end for
end for
```

Рис. 2. Алгоритм соединения *MHJ*

Алгоритм *MHJ* предполагает, что большее (тестируемое) отношение *R* не помещается в оперативную память одного узла кластера, а меньшее (опорное) отношение *S* целиком помещается в оперативной памяти. Для опорного отношения строится хеш-таблица, которая обеспечивает

высокую скорость доступа к элементам отношения. Затем для каждой строки из тестируемого отношения выполняется поиск значений, соответствующих условию соединения.

Для организации межпроцессорных обменов в параллельных системах баз данных [10] будем использовать оператор *Exchange*. Оператор *Exchange* перераспределяет данные между сопроцессорными узлами в соответствии с некоторой функцией распределения, которая определяет, на каком сопроцессорном модуле должен обрабатываться том.

Пусть нам необходимо смоделировать выполнение параллельной транзакции, представляющее собой естественное соединение двух отношений *R* и *S* по общему атрибуту *A*. Мы предполагаем, что опорное отношение *S* фрагментировано по всем узлам по атрибуту соединения, тогда коэффициент перекоса для отношения *S* будет равен $\omega = 1$. Тестируемое отношение *R* фрагментировано по всем узлам по произвольному атрибуту, значения атрибута *A* равномерно распределены по фрагментам отношения *R*. Для отношения *R* вводим коэффициент перекоса $\omega < 1$. Например, при $\omega = 0.8$ каждый параллельный агент при выполнении соединения отправляет своим контрагентам по сети около 20 % кортежей из своего фрагмента. При этом все контрагенты получат примерно одинаковое количество кортежей. Параллельный алгоритм соединения *MHJ* представлен на рис. 3.

```
отправить отношение S на сопроцессор
#pragma omp parallel for
for each s in S
    вычислить значение хеш-функции для s по атрибуту A
    добавить r в хеш-таблицу
end for
for each r in R
    вычислить значение хеш-функции для r по атрибуту A
    for each s in хеш-таблица
        if join(r, s)   R S
            return (r, s)
    end for
end for
```

Рис. 3. Параллельный алгоритм соединения MHJ

Предположим, что отношения *R* и *S* находятся в сжатом виде и представлены в виде томов. Все тома отношения *R* имеют равные коэффициенты перекоса данных и значения атрибута *A* равномерно распределено в каждом томе. На рис. 4 представлено, как будет выглядеть параллельный алгоритм соединения *MHJ*, оперирующий сжатыми данными.

```
отправить отношение S на сопроцессор
#pragma omp parallel for
for each t<s> in S     //для каждого тома t отношения R
    for each s in t    //для каждого кортежа s тома t
        вычислить значение хеш-функции для s по атрибуту A
        добавить s в хеш-таблицу
    end for
end for
#pragma omp parallel for
for each t<r> in R     //для каждого тома t отношения R
    for each r in t    //для каждого кортежа r тома t
        вычислить значение хеш-функции для r по атрибуту A
        for each s in хеш-таблица
            if Join(r, s)
                return (r, s)
        end for
    end for
end for
```

Рис. 4. Параллельный алгоритм соединения MHJ, оперирующий сжатыми данными

В случае моделирования обработки транзакции в системой баз данных с более чем двумя сопроцессорными модулями необходимо учитывать динамически изменяющуюся нагрузку на

сетевой коммутатор, которая может вызывать задержку передачи данных, а также нагрузку на шину PCI-Express, через которую подключаются многоядерные сопроцессоры к процессорному модулю. В связи с этим было принято решение оценивать эффективность работы моделируемой системы методами имитационного моделирования при помощи создания программного эмулятора мультипроцессоров баз данных, оснащенных многоядерными ускорителями.

Программный эмулятор. На основе предложенной модели мультипроцессоров баз данных, оснащенных многоядерными сопроцессорами разработан эмулятор, позволяющий оценить применимость методов сжатия при передаче данных между сопроцессором и основной памятью. Один такт работы модели представлен на рис. 5.

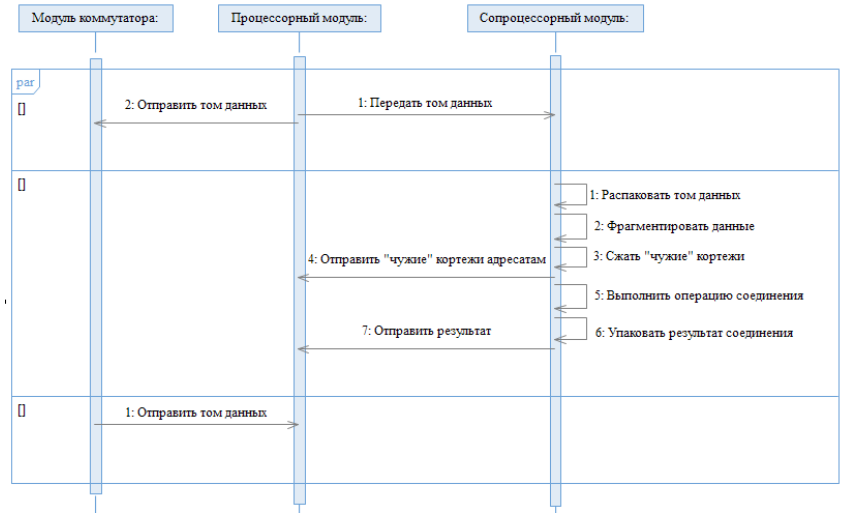

Рис. 5. Такт работы модели

Основные принципы работы эмулятора:

- Сопроцессорный модуль за один такт обрабатывает один том данных.
- Процессорный модуль передает необработанный том данных на сопроцессорный модуль. Если в буфере процессорного модуля есть том для передачи на сопроцессор, то он отправляет этот том в очередь модуля коммутатора либо в очередь своего сопроцессора.
- Модуль коммутатора извлекает из своей очереди тома данных и отправляет их получателям – в буфер процессорного модуля.

Для подтверждения адекватности разработанного эмулятора были проведены вычислительные эксперименты. Предполагалось, что отношения R и S имеют общий атрибут A. Отношение S фрагментировано по общему атрибуту, отношение R фрагментировано по всем узлам по произвольному атрибуту. Процентное содержание «своих» кортежей во фрагментах отношения R определялось коэффициентом перекоса ω. На рис. **6** приведены результаты экспериментов, в ходе которых на суперкомпьютере «Торнадо ЮУрГУ» вычислялось естественное соединение отношений R и S с использованием алгоритма MHJ с различным значением перекоса.

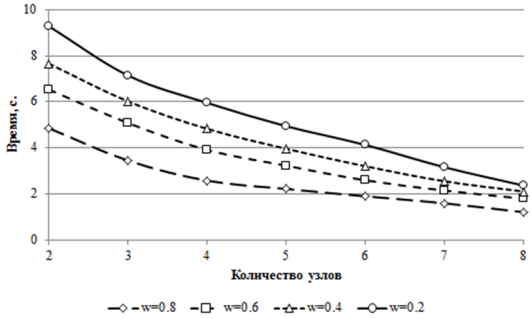

Рис. 6. Выполнение операции соединения на суперкомпьютере «Торнадо ЮУрГУ»

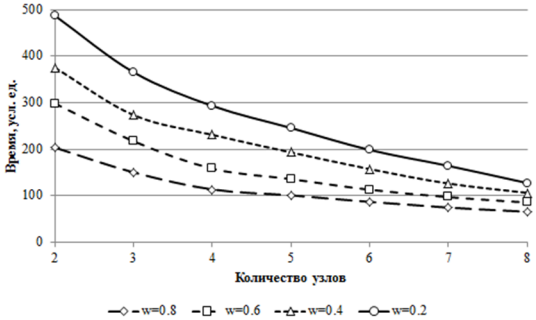

Рис. 7. Моделирование выполнения запроса на эмуляторе

Далее с помощью разработанного эмулятора было произведено моделирование выполнения алгоритма MHJ на HDM- дереве, описывающем архитектуру суперкомпьютера «Торнадо ЮУрГУ». Результаты экспериментов приведены на рис. 7.

Сравнение графиков на рис. 6 и рис. 7 показывает, что эмулятор адекватно моделирует выполнение транзакции на вычислительном кластере. Это подтверждает адекватность модели.

Пример использования эмулятора. Исследовалось влияние пропускной способности системной шины на общую производительность вычислительного узла с многоядерным сопроцессором, а также эффективность использования нескольких сопроцессоров на одном узле. Эксперименты проводились для различных значений пропускной способности коммуникационной сети, шины PCI-Express. Результаты экспериментов приведены на рис. 8.

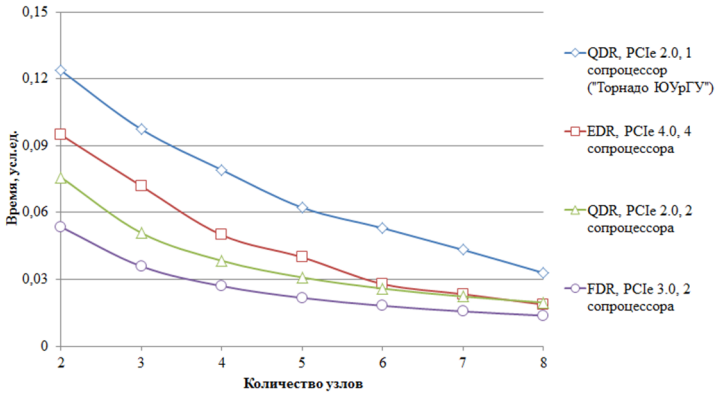

Рис. 8. Влияние пропускной способности системной шины узла на ускорение

В результате данного эксперимента наибольшую эффективность показала архитектура с коммуникационной сетью InfiniBand FDR, системной шиной PCI-Express 3.0 и двумя сопроцессорами на узле.

Применимость сжатия данных. Интерес к задаче сжатия данных в СУБД изначально был обусловлен стремлением уменьшить объем баз данных, так как цена подсистемы ввода-вывода составляла основную часть стоимости аппаратуры. В настоящее время фокус интереса все больше смещается на увеличение производительности системы управления данными за счет использования сжатия.

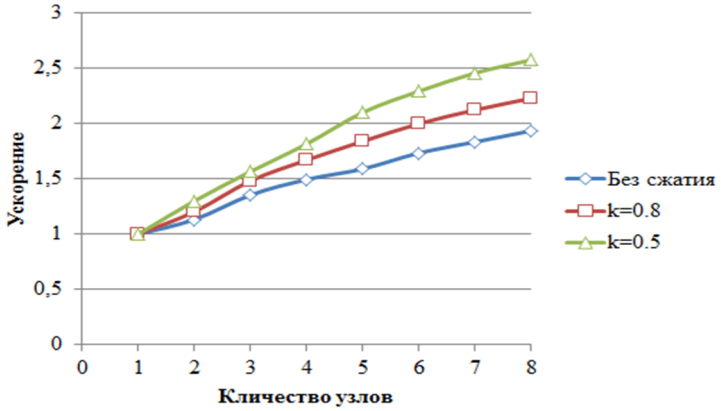

Рис. 9. Влияние коэффициента сжатия на время выполнения транзакции

Для оценки применимости сжатия было исследовано влияние коэффициента сжатия данных на ускорение (рис. 9). В результате анализа графика можно сделать вывод, что применение сжатия данных в многопроцессорных системах баз данных, построенных на основе гибридных вычислительных узлов с многоядерными сопроцессорами, эффективно и является перспективным направлением развития параллельных систем баз данных.

Заключение. В работе рассмотрены вопросы моделирования и анализа многопроцессорных систем баз данных, оснащенных многоядерными сопроцессорами. В результате работы получены следующие результаты: разработана модель, позволяющая моделировать многопроцессор-

ные системы баз данных, оснащенные многоядерными сопроцессорами; на основе предложенной модели разработан программный эмулятор, позволяющий моделировать и исследовать различные многопроцессорные конфигурации; проведены вычислительные эксперименты, направленные на поиск оптимальных аппаратных архитектур мультипроцессоров баз данных; произведена оценка применимости сжатия при передаче данных между сопроцессором и основной памятью. Дальнейшим направлением работ будет расширение модели на СУБД с колоночным хранением данных [5].

Литература

1. Kim C., Chhugani J. Designing Fast Architecture-Sensitive Tree Search on Modern Multi-core/Many-Core Processors // ACM Trans. Database Syst. USA: ACM, 2010. Vol. 36. No. 4. Art. 22.
2. Sokolinsky L.B. Survey Of Architectures of Parallel Database Systems // Programming and Computer Software. 2004. Т. 30. № 6. С. 337-346.
3. Беседин К.Ю. Костенецкий П.С. Исследование эффективности различных методов сжатия при передаче данных из основной памяти в память сопроцессора Intel Xeon Phi // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров: Труды Международной суперкомпьютерной конференции (22-27 сентября 2014 г., Новороссийск). М.: Изд-во МГУ, 2014. С. 114-119.
4. Беседин К.Ю., Костенецкий П.С. Моделирование обработки запросов на гибридных вычислительных системах с многоядерными сопроцессорами и графическими ускорителями // Программные системы: теория и приложения: электрон. научн. журн. Института программных систем им. А.К. Айламазяна РАН. 2014. Т. 5, № 1(19), С. 91–110
5. Иванова Е.В., Соколинский Л.Б. Использование распределенных колоночных индексов для выполнения запросов к сверхбольшим базам данных // Параллельные вычислительные технологии (ПаВТ'2014): труды международной научной конференции (1–3 апреля 2014 г., г. Ростов-на-Дону). Челябинск: Издательский центр ЮУрГУ, 2014. С. 270–275.
6. Костенецкий П.С., Беседин К.Ю. Исследование эффективности различных методов сжатия при передаче данных из основной памяти в память сопроцессора Intel Xeon Phi // Вычислительные методы и программирование. 2014. Т. 15 № 4 С. 593-601.
7. Костенецкий П.С., Соколинский Л.Б. Моделирование параллельных систем баз данных: учебное пособие. Челябинск: Фотохудожник, 2012. 78 с.
8. Костенецкий П.С., Соколинский Л.Б. Моделирование иерархических многопроцессорных систем баз данных // Программирование. Москва: МАИК “Наука/Интерпериодика”. Т. 39 № 1. 2013.
9. Осипова А.М., Костенецкий П.С. Моделирование мультипроцессоров систем баз данных с многоядерными ускорителями // Научный сервис в сети Интернет: все грани параллелизма: Труды Международной суперкомпьютерной конференции (23-28 сентября 2013 г., Новороссийск). М.: Изд-во МГУ, 2013. С. 194.
10. Пан К.С., Цымблер М.Л. Разработка параллельной СУБД на основе последовательной СУБД PostgreSQL с открытым исходным кодом // Вестник ЮУрГУ. Серия "Математическое моделирование и программирование". 2012. № 18(277). Вып. 12. С. 112–120.
11. Приказчиков С.О., Костенецкий П.С. Применение графических ускорителей для обработки запросов над сжатыми данными в параллельных системах баз данных // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров: Труды Международной суперкомпьютерной конференции (22-27 сентября 2014 г., Новороссийск). М.: Изд-во МГУ, 2014. С. 277-279.
12. Соколинский Л.Б. Параллельные машины баз данных // Природа. 2001. № 8. С. 10-17.