БАЛАНСИРОВКА ЗАГРУЗКИ ПРИ ВЫПОЛНЕНИИ ОПЕРАЦИЙ СОЕДИНЕНИЯ В ПАРАЛЛЕЛЬНЫХ СУБД ДЛЯ КЛАСТЕРНЫХ СИСТЕМ*

А.В. Лепихов

В настоящее время большое распространение получили относительно недорогие кластерные системы, которые отличаются простотой наращивания вычислительных ресурсов (памяти, дисков, процессоров) и потенциально обладают неограниченной производительностью. В рейтинге TOP500 самых мощных компьютеров мира доля кластерных систем составляет 80%. В соответствии с этим становится актуальной задача разработки новых параллельных приложений и адаптации существующих приложений к кластерным системам.

В области параллельных систем баз данных одним из наиболее перспективных подходов к созданию параллельных СУБД является метод *инкапсуляции параллелизма*. Данный подход предполагает создание параллельной СУБД путем внедрения специальных операторов-капсул *exchange* в последовательный план выполнения запроса. Изначально метод инкапсуляции параллелизма был разработан для многопроцессорных систем с общей памятью [1]. Затем он был обобщен для мультипроцессоров с массовым параллелизмом и для кластерных систем [2].

Основной проблемой для кластерных систем является проблема перекосов [3]. Эффект перекоса состоит в неравномерной загрузке процессорных узлов. Основной причиной возникновения перекосов в системах баз данных является неравномерное распределение данных по процессорным узлам. Перекосы могут приводить к полной деградации общей производительности системы [4].

В данной работе исследуется метод *частичного зеркалирования* [5], который может быть эффективно использован для балансировки загрузки в кластерных системах. Предлагается оригинальный алгоритм балансировки загрузки, основанный на использовании метода частичного зеркалирования и оператора *exchange*. Описывается реализация этого алгоритма для операции соединения хешированием в основной памяти. Данный алгоритм реализован в прототипе параллельной СУБД Омега. Исследована эффективность предложенного алгоритма балансировки в условиях перекосов в распределении данных по процессорным узлам.

Метод частичного зеркалирования. В основе метода частичного зеркалирования лежит *функция фрагментации* [2] φ_X, $\varphi_X : X \to N$, где X – произвольное отношение. Для любого $x \in X$ значение $\varphi_X(x)$ определяет номер узла, на котором хранится кортеж *x*.

Через X_k будем обозначать фрагмент отношения X, хранящийся на *k*-том узле: $X_k = \{x \mid x \in X, \varphi_X(x) = k\}$. Через X_k^m будем обозначать реплику X_k, хранящуюся на *m*-том узле. В частности, имеем $X_k^k = X_k$.

Мы предполагаем, что каждое отношение фрагментировано по K узлам. Мы также предполагаем, что каждый фрагмент реплицирован (возможно, частично) на M узлах. Нумерация узлов начинается с единицы. Фрагмент состоит из набора *сегментов* – блоков кортежей заданного размера. Размер реплики *k*-того фрагмента на *m*-том узле задается *коэффициентом репликации* ρ_k^m. Поскольку отношение *S* значительно больше отношения *R* то мы будем предполагать, что балансировка выполняется только для отношения *S*.

Балансировка происходит с точностью до сегмента. При этом в балансировке, в качестве рабочих узлов, могут принимать участие только те узлы, на которых имеются фрагменты исходных отношений.

Опишем механизм балансировки загрузки для произвольной унарной однопроходной операции над отношением S. Схема работы предлагаемого алгоритма изображена на рис. 1 на примере кластера с двумя процессорными узлами и коэффициентом репликации 0.5.

Пусть процессоры P_1 и P_2 выполняют выборку кортежей из отношения *S*. Фрагмент S_1 назначается процессору P_1, фрагмент S_2 – процессору P_2. Пусть в момент времени t_2 процессор P_1 уже обработал выделенные ему фрагменты, в то время как процессор P_2 выполнил только часть назначенной ему работы.

В этом случае происходит перераспределение оставшейся необработанной части фрагмента S_2. Причем процессор P_1 использует для обработки реплику S_2^1. Процесс продолжается до тех пор, пока тестируемое отношение S не будет полностью обработано. Алгоритм очевидным образом обобщается на произвольное количество процессоров. Перераспределение необработанной части фрагмента может быть произведено заново, если к моменту времени t_4 процессор P_2 не завершит обработку своего фрагмента.

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 06-07-89148).

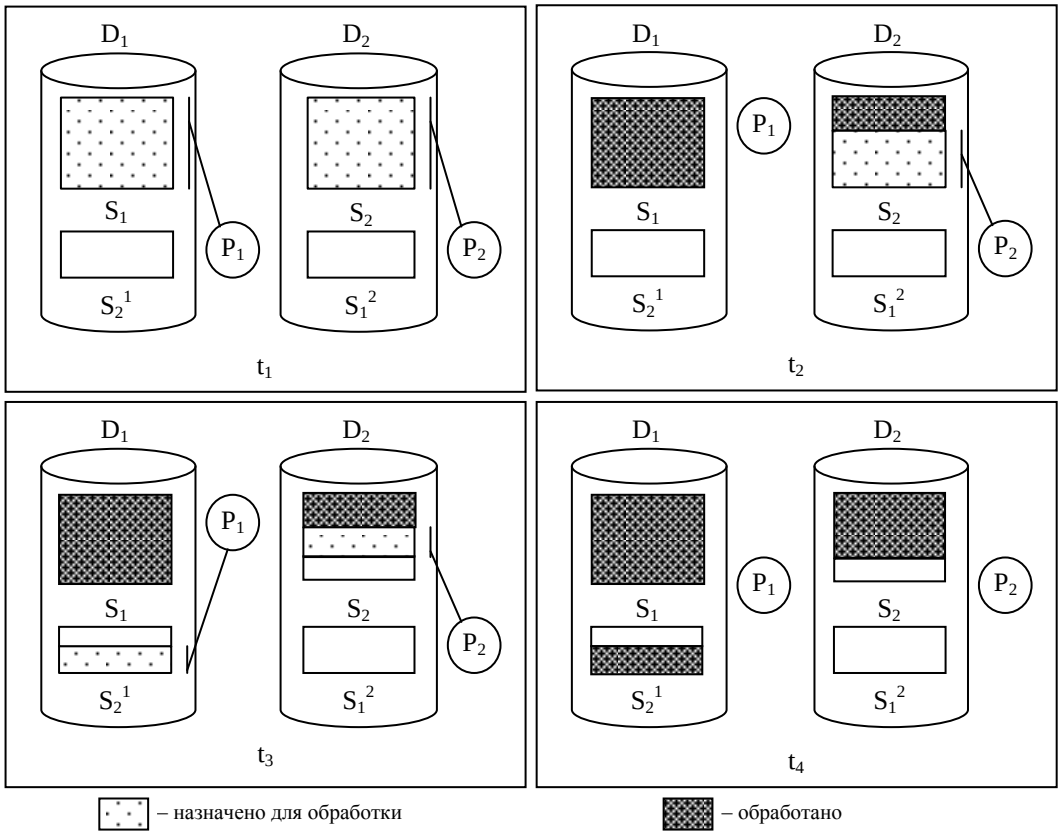

Рис. 1. Балансировка загрузки для двух узлов с $\rho=0.5$.

Соединение хешированием в основной памяти. Современные кластерные системы имеют значительную суммарную оперативную память. Например, кластер СКИФ Урал [8] имеет суммарную оперативную память 1.33 ТБ. В контексте систем баз данных это означает, что при использовании фрагментного параллелизма отношение размером меньше 1.33 ТБ может быть фрагментировано по процессорным узлам таким образом, что каждый фрагмент этого отношения целиком поместится в оперативную память. В бинарных реляционных операциях обычно одно из отношений по размеру значительно превосходит другое. Применительно к операции соединения меньшее отношение называется *опорным*, а большее - *тестируемым*. В реальных приложениях баз данных случаи, когда опорное отношение имеет размер более терабайта, достаточно редки. Поэтому в кластерных системах для операции соединения может быть использован простой и эффективный алгоритм соединения методом хеширования в основной памяти [6].

Для реализации балансировки загрузки в набор операций реляционной алгебры добавляется специальный оператор **exchange** [2]. Оператор **exchange** выполняет перераспределение кортежей между процессорными узлами в соответствии с *функцией распределения*, которая для каждого кортежа определяет номер процессорного узла, на котором данный кортеж должен быть обработан. Пример использования оператора **exchange** показан на рис. 2. На данном рисунке представлен параллельный план запроса, реализующего операцию **MHJ** соединения хешированием в основной памяти. Оператор **scan** выполняет сканирование отношения. Оператор **exchange** вставляется в качестве левого и правого сына оператора **MHJ**. В процессе обработки запроса он выполняет перераспределение кортежей, поступающих от оператора **scan**, между процессорными узлами.

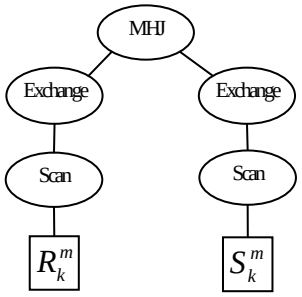

Рис. 2. Параллельный план запроса R▷◁S

Общую схему обработки операции соединения хешированием в основной памяти можно описать следующим образом. Параллельный план (см. рис. 2) передается на каждый процессорный узел, содержащий фрагмент базы данных. Параллельный агент, расположенный на узле, выполняет данный параллельный план.

Процесс выполнения соединения можно разделить на две фазы. На первой фазе выполняется инициализация операции соединения. Кортежи опорного отношения перераспределяются между процессорными узлами. Параллельный агент строит фрагмент хеш-таблицы опорного отношения в основной памяти в соответствии с некоторой хеш-функцией.

На второй фазе выполняется обработка тестируемого отношения. Кортежи тестируемого отношения перераспределяются между узлами, и выполняется операция соединения.

Операция формирования хеш-таблицы не требует балансировки ввиду небольшого размера опорного отношения. Таким образом, при выполнении операции соединения методом хеширования в основной памяти балансировка используется на этапе обработки тестируемого отношения и сводится к балансировке загрузки унарной однопроходной операции, описанной выше.

Результаты экспериментов. Метод балансировки загрузки реализован в прототипе параллельной СУБД «Омега» [7]. На базе данного прототипа проведены вычислительные эксперименты с целью оценки эффективности предложенного метода балансировки при выполнении операции соединения и оценка оптимального значения коэффициента репликации ρ.

Испытания проводились на вычислительном кластере ЮУрГУ – СКИФ Урал [8]. Основные системные параметры кластера приведены в табл. 1.

В экспериментах использовалась тестовая база данных, состоящая из отношений с целочисленными атрибутами. Для генерации значений атрибутов использовалось распределение «80-20», «45-20» и нормальное распределение. По умолчанию, все атрибуты генерировались в соответствии с распределением «80-20» согласно которому, двадцати процентам значений атрибута соответствует восемьдесят процентов кортежей [9].

Табл. 1. Параметры системы баз данных

Параметр	Значение
Параметры кластера	
Количество процессорных узлов	166
Количество дисков на процессорном узле	1
Частота ядра процессора	3.0 GHz
Оперативная память	1.33 ТБ
Дисковая память	49.29 ТБ
Пропускная способность сети	20 Гбит/сек
Параметры базы данных	
Размер отношения R	0.4 ГБ
Размер отношения S	8 ГБ
Размер сегмента	2 МБ

В экспериментах исследовалось влияние механизма балансировки на время выполнения операции соединения хешированием в основной памяти. В ходе эксперимента прототип «Омега» запускался на 20 процессорных узлах кластера СКИФ Урал с различным коэффициентом репликации ρ. Коэффициент репликации варьировался в пределах от 0 (отсутствие репликации) до 1 (полная копия базы данных на каждом узле). Эксперименты показали, что при использовании механизма балансировки загрузки наблюдается значительное увеличение скорости выполнения операции соединения. Результаты данного эксперимента изображены на рис. 3.

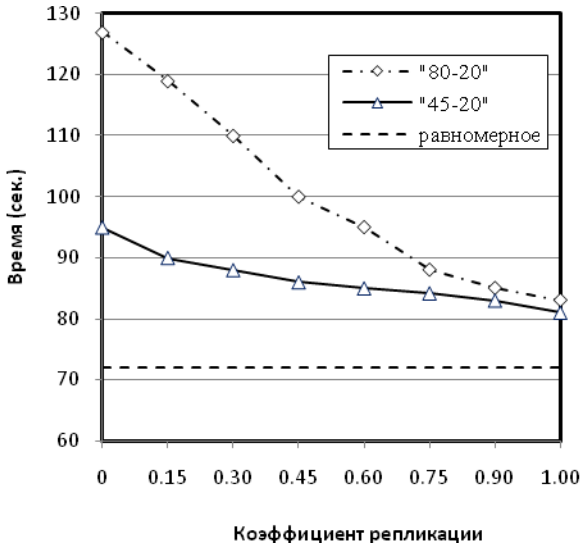

Рис **3**. Зависимость времени выполнения запроса от коэффициента репликации ρ.

Наибольший прирост производительности возникает при выполнении запросов над базой данных, сгенерированной и фрагментированной в соответствии с правилом «80-20». Коэффициент репликации 0.15 дает прирост производительности 6%, При ρ=0.60, прирост производительности составляет 25%, а при пол-

ной репликации (ρ=1.00) – 35%. Таким образом, в конкретных приложениях баз данных коэффициент репликации можно варьировать с целью уменьшения накладных расходов на поддержание реплик.

В настоящее время ведутся работы по реализации метода балансировки для алгоритмов GRACE-соединения и гибридного соединения. Планируется внедрение предложенного метода балансировки загрузки в СУБД с открытым исходным кодом PostgreSQL.

ЛИТЕРАТУРА

1. *Graefe G.* Encapsulation of Parallelism in the Volcano Query Processing Systems // Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990. ACM Press. -1990. -P. 102-111.
2. *Соколинский Л.Б.* Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой // Программирование. -2001. -No. 6. -С. 13-29.
3. *Maertens, H.* A Classification of Skew Effects in Parallel Database Systems // Proceedings of 7th International Euro-Par Conference, August 28–31, 2001, Manchester, UK, P. 291–300.
4. *Lakshmi M.S., Yu P.S.* Effectiveness of Parallel Joins // IEEE Transactions on Knowledge and Data Engineering. -1990. -Vol. 2, No. 4. -P. 410-424
5. *Костенецкий П.С., Лепихов А.В., Соколинский Л.Б.* Технологии параллельных систем баз данных для иерархических многопроцессорных сред // Автоматика и телемеханика. -2007. -Том 68, №5. -С. 847-859.
6. *DeWitt D.J., Gerber R.H.* Multiprocessor Hash-Based Join Algorithms // VLDB'85, Proceedings of 11th International Conference on Very Large Data Bases, August 21-23, 1985, Stockholm, Sweden. Morgan Kaufmann. – 1985. –P. 151–164.
7. Прототип параллельной СУБД «Омега»: [http://omega.susu.ru]
8. Высокопроизводительный вычислительный кластер «СКИФ Урал»: [http://skif-ural.susu.ac.ru]
9. *Heising W.P.* Note on Random Addressing Techniques // IBM System Journal. –1963. –Vol. 2, No. 2. -P. 112-116.