МОДЕЛИРОВАНИЕ ПАРАЛЛЕЛЬНЫХ СИСТЕМ БАЗ ДАННЫХ ДЛЯ ВЫЧИСЛИТЕЛЬНЫХ КЛАСТЕРОВ

П.С. Костенецкий

1. Введение

В настоящее время наиболее распространенным классом многопроцессорных систем стали вычислительные кластеры. Это связано с появлением на рынке высокоэффективных типовых решений демонстрирующих наилучшее соотношение «производительность/цена». На вычислительных кластерах решается широкий спектр задач. Используются они и для приложений баз данных. В соответствие с этим актуальной является задача исследования новых кластерных архитектур в контексте оперативной обработки транзакций (OLTP). Для этих целей была разработана модель DMM (Database Multiprocessor Model), позволяющая моделировать и исследовать произвольные кластерные конфигурации. Модель DMM отличается от других известных моделей для многопроцессорных систем, таких как BSP (Bulk Synchronous Parallel) [1., 2.] и H-BSP (Hierarchical Bulk Synchronous Parallel) [3.], тем, что она учитывает специфику приложений параллельных баз данных, характеризующуюся тем, что основное время при выполнении запроса расходуется на операции обменов с дисками и передачу данных по соединительной сети. Объем процессорных вычислений при этом остается пренебрежимо малым.

Модель DMM включает в себя модель аппаратного обеспечения, модель операционной среды, стоимостную модель и модель транзакций. Модель аппаратного обеспечения, операционной среды и стоимостная модель были рассмотрены в статье [4.]. В настоящей работе предлагается модель транзакций, описывается реализация модели DMM в виде эмулятора виртуальных мультипроцессоров баз данных DMS (Database Multiprocessor Simulator) и приводятся результаты вычислительных экспериментов, выполненных с использованием эмулятора.

2. Модель DMM

В модели DMM многопроцессорная система представляется в виде *DM*-дерева. *DM-дерево* – это ориентированное дерево, узлы которого относятся к одному из трех классов:

1. процессорные модули;
2. дисковые модули;
3. модули сетевых концентраторов.

Дуги дерева соответствуют потокам данных. *Процессорные модули* являются абстрактным представлением реальных процессорных устройств. *Дисковые модули* представляют накопители на жестких магнитных дисках. *Модули сетевых концентраторов* используются для представления произвольного *интерконнекта* (соединительной сети), объединяющего различные процессорные и дисковые устройства. В качестве интерконнекта могут фигурировать как отдельные сетевые устройства (коммутатор, концентратор и др.), так и системная шина, соединяющая процессор с периферийными устройствами. Модель DMM не предусматривает представление модулей оперативной памяти, так как в задачах OLTP время взаимодействия процессоров и оперативной памяти несоизмеримо меньше времени обменов данными с внешними устройствами. Пример *DM*-дерева представлен на рис. 1.

На рис. 1 представлена модель простейшего вычислительного кластера, узлы которого состоят из одного процессора P_i и одного жесткого диска D_i соединенных системной шиной H_i. Все вычислительные узлы

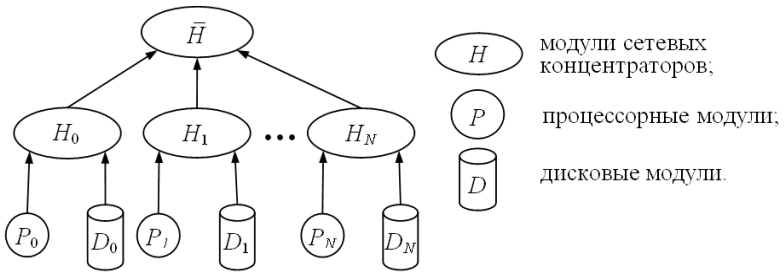

Рис. 1. Пример DM-дерева.

кластера объединяются общим интерконнектом $\bar{H}$.

На структуру *DM*-дерева накладываются следующие ограничения:

1. корнем *DM*-дерева может быть только модуль сетевого концентратора;

2. дисковые и процессорные модули не могут иметь дочерних узлов, то есть они всегда являются листьями *DM*-дерева.

В модели DMM время представляется в виде дискретных промежутков, называемыми тактами. За каждый такт любое виртуальное устройство может выполнить не более одной операции.

3. Модель транзакций

DMM модель поддерживает как последовательные, так и параллельные транзакции.

Последовательная транзакция – это транзакция, которая выполняется на одном процессорном модуле. Последовательная транзакция *Z* моделируется путем задания двух групп процессов *r* и *w*: *Z*={*r*,*w*}, *r* ▪ *w* ▪ ▪. Группа *r* включает в себя *читающие* процессы, группа w – *пишущие* процессы. Процессы абстрагируют операции чтения/записи, выполняемые при обработке транзакций.

Для каждого читающего и пишущего процесса задается *количество обращений к диску*, которое он должен произвести. После того, как процесс выполнил все обращения, он удаляется из множества *r* или *w* соответственно. Транзакция *Z*={*r*, *w*} считается завершенной, когда *r* ▪ *w* ▪ ▪.

В модели DMM каждая транзакция *Z*={*r*, *w*} разбивается на конечную последовательность шагов: Z_1, ...,Z_s. Здесь *s* обозначает количество шагов транзакции. Каждый шаг x_i (*i* = 1,…,*s*) описывается тройкой чисел (n_i, p_i, d_i), где d_i – номер диска, с которым процесс *x* обменивается данными на *i*-том шаге; n_i – количество обращений к диску; p_i – вероятность обращения процесса *x* к диску с номером d_i на каждом такте работы эмулятора в течении *i*-го шага.

Параллельная транзакция – это транзакция, для выполнения которой используются два или более процессорных модулей. Параллельная транзакция *Z*, выполняемая на *l* процессорных модулях, моделируется путем задания *l* последовательных транзакций Z^1,…, Z^l, каждая из которых выполняется на отдельном процессорном модуле. Таким образом, мы можем считать, что в системе выполняются только последовательные транзакции.

Опишем алгоритм, моделирующий выполнение отдельной транзакции на процессорном модуле. Пусть задана транзакция *Z* = {*r, w*}, состоящая из *s* шагов: Z_1,…, Z_s. Обозначим через *X* множество всех читающих и пишущих процессов, составляющих транзакцию *Z*: *X* = *r* ▪ *w*. Пусть *X*={x^1,…, x^r}. Каждый процесс x^j (*j* = 1, …, *r*) делится на *s* шагов: $x_1^j, \ldots, x_s^j$. Пусть выполнение транзакции *Z* находится на *i*-том шаге. Предположим, что процесс x^j *получил управление*. Пусть *i*-тый шаг процесса x^j имеет вид $x_i^j=(n_i^j, p_i^j, d_i^j)$. Предполагается, что процесс x^j может получить управление на *i*-том шаге только в случае $n_i^j > 0$. Тогда на текущем такте выполняется следующая последовательность действий.

1. Значение n_i^j уменьшается на единицу.
2. Процесс x^j инициирует операцию обмена с диском, имеющим номер d_i^j .
3. Если $\sum_{j=1}^{r} n_i^j = 0$ и *i*<*s*, то увеличить *i* (номер текущего шага транзакции) на единицу.

Состояние n_i^j соответствует завершению *i*-того шага процесса x^j. Состояние $\sum_{j=1}^{r} n_i^j = 0$ соответствует завершению *i*-того шага транзакции *Z*. Таким образом, переход выполнения транзакции на шаг *i*+1 происходит только тогда, когда все входящие в нее процессы завершили выполнение *i*-того шага.

Модель DMM допускает выполнение на одном процессоре смеси последовательных транзакций {Zi | i = 1,…, k} в режиме разделения времени. При этом каждая транзакция Zi (i = 1,…, k) представляется своей собственной парой групп читающих и пишущих процессов: Zi={ri, wi}. Все множество процессов моделирующих выполнение смеси транзакций на некотором процессоре P □ □ определяется следующим образом:

$$\Phi_P = \bigcup_{i=1}^{k} (\rho^i \cup \omega^i).$$

При этом, при запуске новой транзакции на процессоре P, в множество Фр динамически добавляются читающие и пишущие процессы, представляющие данную транзакцию. Если какой-либо процесс завершается, то он удаляется из множества Фр.

Рассмотрим как в модели DMM организуется выполнение смеси транзакций Фр на некотором процессоре P. При этом мы предполагаем, что процессор P не находится в состоянии ожидания (процессор переходит в состояние ожидания, если количество незавершенных обменов, инициированных этим процессором, превысило некоторую фиксированную величину). На каждом такте работы эмулятора процессор P может инициализировать одну операцию обмена с дисками. В соответствие с этим процессор должен выбрать некоторый процесс x ▪ Фр и произвести операцию чтения с диска или записи на диск, ассоциированный с x на текущем шаге. Такой процесс называется активным. Все процессы из множества Фр организуются в связный список. Для определения активного процесса используется алгоритм, изображенный на Рис. 2.

```
/* Процедура выбора активного процесса x*/
for (P = P.begin(); P != P.end(); P++){ // Цикл по процессорам
   prob = 0; // Суммарная вероятность
   for (x = P.Ф.begin(); x != P.Ф.end(); x++){ // Цикл по процессам
      if (n(x,i(x)) == 0) continue;
      prob += p(x,i(x));
      if (g(prob)) return x;
   };
};
```

Рис. 2. Процедура выбора активного процесса *x*.

Здесь P – множество процессорных модулей; s(x) – общее количество шагов процесса x; i(x) – номер текущего шага процесса x; n(x,i) – количество обращений к диску, которые осталось выполнить процессу x на шаге i; p(x,i) – вероятность обращения процесса x к диску при выполнении *i*-го шага; g – функция срабатывания, вычисляемая следующим образом. Для каждого шага *i* процесса *x* известна вероятность p_i^x обращения процесса *x* к диску, ассоциированному с этим процессом на *i*-том шаге. Функция срабатывания $g(p_i^x)=G$ определяется как функция дискретной случайной величины *G*, закон распределения которой задается следующим рядом распределения:

g	1	0
p	p_i^x	$1-p_i^x$

4. Формирование транзакций

Рассмотрим формирование транзакций в модели DMM на примере алгоритма *MHJ* (*Main memory Hash Join*) – *соединение хешированием в основной памяти*. Данный алгоритм интенсивно используется в современных СУБД при обработке запросов в тех случаях, когда одно из соединяемых отношений целиком помещается в оперативной памяти. Алгоритм MHJ делится на фазы *построения* и *сравнения*. На *фазе построения* строится хеш-таблица опорного отношения *R* в оперативной памяти. При этом каждый процессорный узел выполняет следующие операции:

1. Покортежное сканирование своего фрагмента отношения *R*. Для каждого кортежа вычисляется значение функции распределения ψ, вырабатывающей номер узла-приемника данного кортежа. Кортеж передается на узел-приемник.
2. Строится хеш-таблица в оперативной памяти при помощи функции хеширования h(t), используя кортежи, переданные ему на шаге 1.

На *фазе сравнения* выполняется обработка тестируемого отношения *S* и соединение с кортежами отношения *R*. При этом каждый процессорный узел выполняет следующие операции:

1. Выполняет покортежное сканирование своего фрагмента отношения *S*. Для каждого кортежа вычисляется значение функции распределения ψ. Кортеж передается на узел-приемник.
2. Принимает переданный ему на шаге 1 кортеж и выполняет соединение с кортежами из хеш-таблицы, построенной на шаге 2 фазы распределения. Формирует результирующий кортеж.

Перераспределение кортежей между процессорными узлами на каждой фазе алгоритма осуществляет оператор exchange. На рис. 3. представлен параллельный план запроса, реализующего операцию MHJ. Оператор scan выполняет сканирование отношения. Оператор exchange вставляется в качестве левого и правого сына оператора MHJ и в процессе обработки запроса выполняет перераспределение кортежей, поступающих от оператора scan, между процессорными узлами.

Пусть нам необходимо сформировать модель параллельной транзакции, представляющей собой естественное соединение двух отношений *R* и S по общему атрибуту *A*. Мы предполагаем, что значения атрибута *A* равномерно распределены по фрагментам отношений *R* и *S*. Для обоих отношений мы вводим коэффициент перекоса *m*, в соответствии с которым кортежи подразделяются на два класса: «свои» и «чужие». Коэффициент *m* указывает долю «своих» кортежей во фрагменте. «Свои» кортежи должны быть обработаны на своем вычислительном узле. «Чужие» должны быть переданы для обработки на другие вычислительные узлы на основе равномерного распределения. Например, при $m = 0.5$ отношение формируется таким образом, что каждый параллельный агент при выполнении соединения передает своим контрагентам по сети около 50% кортежей из своего фрагмента. При этом все контрагенты получат примерно одинаковое количество кортежей.

Мы предполагаем, что отношение *R* разбивается на равновеликие фрагменты R_i ($i = 0,..., N-1$), каждый из которых хранится на отдельном вычисли тельном узле, номер которого совпадает с номером фрагмента. При этом любой фрагмент R_k может быть целиком размещен в оперативной памяти вычислительного узла. Аналогично, *S* разбивается равновеликие фрагменты S_i ($i = 0,..., N-1$), каждый из которых хранится на отдельном вычисли тельном узле, номер которого также совпадает с номером фрагмента.

Параллельная транзакция представляется как совокупность однотипных последовательных транзакций, каждая из которых выполняется на отдельной вычислительном узле. Пусть на нулевом вычислительном узле (в контексте рис. 1) выполняется такая последовательная транзакция $Z = \{r\}$. Транзакция Z и соответственно все входящие в нее процессы разбиваются на два шага. Первый шаг моделирует фазу построения, второй – фазу сравнения. Множество читающих процессов транзакции Z имеет вид:

$r = \{a^0, \ldots, a^{N-1}, b^0, \ldots, b^{N-1}\}$.

Процессы $a^0, \ldots, a^{N-1}$ соответствуют фазе построения. Процесс a^0 моделирует чтение «своих» кортежей отношения R со своего диска D_0. Он состоит из двух шагов: $a^0 = \{a_1^0, a_2^0\}$, где $a_1^0 = (\mu |R|/N, \mu, 0)$, $a_2^0 = (0,0,0)$. Процессы a^j ($j = 1,..., N-1$) моделируют чтение «своих» кортежей отношения R с чужих дисков. Каждый такой процесс также состоит из двух шагов: $a^j = \{a_1^j, a_2^j\}$, где $a_1^j = ((1-\mu)|R|/(N^2-N), (1-\mu)/(N-1), j)$, $a_2^j = (0,0,j)$.

Процессы $b^0, \ldots, b^{N-1}$ соответствуют фазе сравнения. Процесс b^0 моделирует чтение «своих» кортежей отношения S со своего диска D_0. Он состоит из двух шагов: $b^0 = \{b_1^0, b_2^0\}$, где $b_1^0 = (0,0,\cdot)$, $b_2^0 = (\mu |S|/N, \mu, \cdot)$. Процессы b^j ($j = 1,..., N-1$) моделируют чтение «своих» кортежей отношения S с чужих дисков. Каждый такой процесс также состоит из двух шагов: $b^j = \{b_1^j, b_2^j\}$, где $b_1^j = (0,0,0)$, $b_1^j = ((1-\mu)|S|/(N^2-N), (1-\mu)/(N-1), j)$. Аналогичным образом формируются транзакции, выполняемые на остальных процессорных узлах.

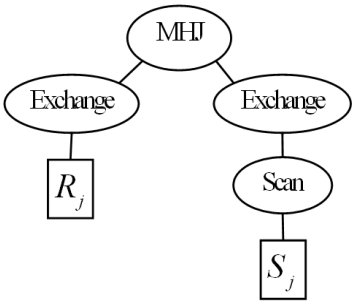

Рис. 3. Параллельный план запроса MHJ.

5. Вычислительные эксперименты

На основе модели DMM на языке C++ был написан эмулятор виртуальных мультипроцессоров баз данных DMS. В данном разделе приводятся результаты вычислительных экспериментов, проведенных с использованием эмулятора и их сравнение с результатами выполнения аналогичных запросов на прототипе параллельной СУБД «Омега» [5.]. На рис. 4 представлен график времени получаемого на эмуляторе DMS, а на рис.5 представлен график времени получаемого на прототипе СУБД «Омега». Эксперименты проводились для различных значений коэффициента m. Эти эксперименты показывают, что DMS адекватно моделирует обработку запроса на вычислительном кластере.

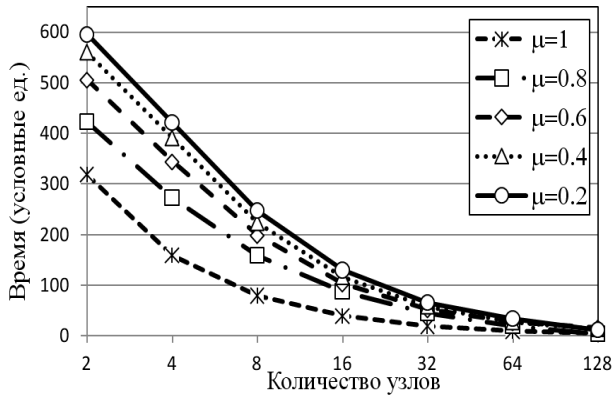

Рис. 4. Моделирование выполнения запроса на DMS.

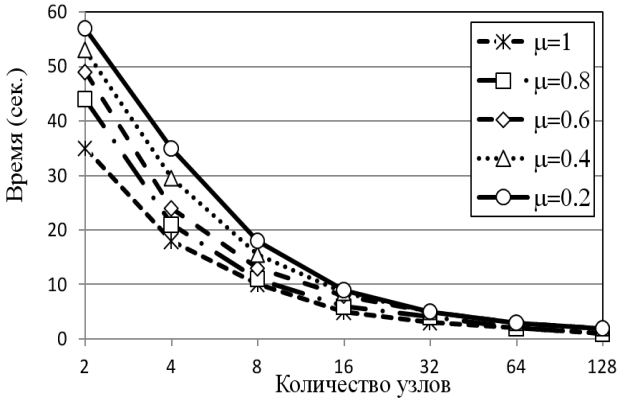

Рис.5. Выполнение запроса на СУБД «Омега».

На рис. 6 представлены графики ускорения при увеличении скорости дисков и коммуникационной сети. Верхний график соответствует ситуации, когда скорость интерконнекта оставалась всегда равной 20 условных единиц, а скорость дисков менялась от 2 до 36 условных единиц. Нижний график соответствует обратной ситуации, когда скорость дисков оставалась всегда равной 20 условных единиц, а скорость интерконнекта менялась от 2 до 36 условных единиц. Проведенные эксперименты позволяют сделать закономерный вывод, что оптимальной является ситуация, когда скорость интерконнекта сравнима со скоростью дисков.

Рис. 6. Зависимость ускорения от скорости коммуникационной сети и дисков.

Заключение

В работе представлена модель DMM, позволяющая моделировать обработку транзакций на многопроцессорных системах. Модель DMM включает в себя модель аппаратного обеспечения, модель операционной среды, стоимостную модель и модель транзакций. Дано описание модели транзакций. Показано, как реальные параллельные транзакции могут быть смоделированы с помощью DMM. На базе модели DMM разработан Эмулятор виртуальных мультипроцессоров баз данных DMS. Эмулятор позволяет исследовать эффективность различных многопроцессорных конфигураций в контексте приложений баз данных класса OLTP. Проведены вычислительные эксперименты показывающие, что во всех случаях эмулятор демонстрирует результаты, идентичные результатам, полученным при работе реальной параллельной СУБД на вычислительном кластере с моделируемой архитектурой. Это доказывает адекватность модели DMM.

Исходные тексты эмулятора свободно доступны в сети Интернет по адресу: http://kps.susu.ru/science/dms/.

ЛИТЕРАТУРА:

1. *Valiant L.G.* A bridging model for parallel computation. Communication of the ACM. 1990. V. 33, No. 8. P. 103–111.
2. *Cormen T.H., Goodrich M.T.* A bridging model for parallel computation, communication, and I/O. ACM Computing Surveys. 1996. V.28, No.4.
3. *Cha H., Lee D.* H-BSP: A Hierarchical BSP Computation Model // The Journal of Supercomputing. 2001. Vol. 18, No. 2. P. 179-200.
4. *Костенецкий П.С., Лепихов А.В., Соколинский Л.Б.* Технологии параллельных систем баз данных для иерархических многопроцессорных сред // Автоматика и телемеханика. 2007. № 5. С. 112-125.
5. Параллельная СУБД «Омега» для многопроцессорных иерархий [Сайт проекта]. URL: http://fireforge.net/projects/omega/ (дата обращения 18.06.2009).