МОДЕЛИРОВАНИЕ ИЕРАРХИЧЕСКИХ АРХИТЕКТУР ПАРАЛЛЕЛЬНЫХ СИСТЕМ БАЗ ДАННЫХ*

П.С. Костенецкий, Л.Б. Соколинский

Введение

В настоящее время все большее распространение получают иерархические архитектуры параллельных систем баз данных [1]. Это связано с тем, что современные многопроцессорные системы в большинстве случаев организуются по иерархическому принципу. Действительно, большая часть суперкомпьютеров сегодня имеют двухуровневую кластерную архитектуру. В соответствии с данной архитектурой многопроцессорная система строится как набор однородных вычислительных модулей, соединенных высокоскоростной сетью. При этом каждый вычислительный модуль является в свою очередь многопроцессорной системой с разделяемой памятью. Другим источником многопроцессорных иерархий являются Grid-технологии [2], позволяющие рассматривать несколько различных суперкомпьютеров как единую вычислительную систему. Подобная Grid-система будет иметь многоуровневую иерархическую структуру. На нижнем уровне иерархии располагаются процессоры отдельных кластерных систем, соединенные высокоскоростной внутренней сетью. На следующем уровне располагаются вычислительные системы, объединенные корпоративной сетью. Внешний уровень иерархии может представлять сеть Интернет.

Иерархические многопроцессорные архитектуры порождают большое количество различных классов конфигураций. Некоторая классификация таких архитектур дана в [1]. В соответствие с этим возникает проблема выбора оптимального класса конфигураций для определенного класса приложений. Данная работа посвящена моделированию многопроцессорных иерархических конфигураций в контексте систем оперативной обработки транзакций (OLTP).

Моделирование всех одноуровневых архитектур для оперативной обработки транзакций было выполнено Стоунбрейкером и Бхайдом [3,4]. В работах [5,6] моделируются некоторые классы двухуровневых конфигураций.

Нами разработана модель *DMM* (*Database Multiprocessor Model*), позволяющая моделировать и исследовать произвольные многопроцессорные иерархические конфигурации в контексте приложений класса OLTP. Модель DMM включает в себя *модель аппаратного обеспечения*, *модель программного обеспечения* и *стоимостную модель*.

Модель аппаратного обеспечения

Аппаратное обеспечение параллельной системы баз данных представляется в виде *DM*-дерева. *DM-дерево* - это дерево, узлы которого относятся к одному из трех классов:

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

Ребра 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*, соединяющего 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, затрачиваемое системой на обмены между процессорами и оперативной памятью, несоизмеримо мало по сравнению с временем, затрачиваемым на обмены с внешними устройствами.

На структуру *DM*-дерева накладываются следующие ограничения:

1) корнем *DM*-дерева может быть только модуль сетевого концентратора;
2) процессорный модуль может быть соединен с дисковым модулем только через модуль сетевого концентратора;
3) дисковые и процессорные модули не могут иметь дочерних узлов, то есть они всегда являются листьями *DM*-дерева;
4) *DM*-дерево должно быть связным, то есть не может быть лесом.

* Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (проект 03-07-90031).

Модель программного обеспечения

В рамках модели DMM наименьшей неделимой единицей обработки данных является пакет. В реальной системе баз данных в качестве пакета может фигурировать один или несколько кортежей. Мы предполагаем, что все пакеты имеют одинаковый размер. Пакет содержит заголовок, включающий в себя адрес отправителя, адрес получателя и другую вспомогательную информацию.

В модели DMM любой процессорный модуль может обмениваться данными с любым дисковым модулем. Поскольку рассматриваются только приложения класса OLTP, мы можем пренебречь накладными расходами на обмены между двумя процессорами через общую оперативную память и затратами на обработку данных внутри процессоров.

С каждым дисковым модулем и модулем сетевого концентра в модели DMM ассоциируется *очередь*, в которую помещаются пересылаемые пакеты.

Модель DMM допускает асинхронный обмен пакетами в том смысле, что процессорный модуль может инициализировать новый обмен, не дожидаясь завершения предыдущего. Однако мы предполагаем, что процессорный модуль может инициализировать не более s_r незавершенных операций чтения и s_w незавершенных операций записи в каждый момент времени.

Время работы системы в модели DMM делится на дискретные промежутки, называемые *тактами*. Такт определяется как фиксированная последовательность шагов, семантика которых будет определена ниже.

Пусть $\mathfrak{P}$ обозначает множество всех процессорных модулей *DM*-дерева, $\mathfrak{D}$ – множество всех дисковых модулей, $\mathfrak{N}$ – множество всех модулей сетевых концентраторов, $\mathfrak{M}=\mathfrak{P}\cup\mathfrak{D}\cup\mathfrak{N}$ – множество всех узлов *DM*-дерева. Для произвольного $M\in\mathfrak{M}$ введем следующие обозначения: $F(M)$ – родительский модуль узла M, $T(M)$ – поддерево с корнем в вершине M.

Процессорный модуль $P\in\mathfrak{P}$ может инициировать операции чтения и записи пакетов. Определим их семантику следующим образом.

Операция чтения. Пусть процессорному модулю P требуется прочитать пакет E с диска $D\in\mathfrak{D}$. Если процессор P ранее инициализировал s_r незавершенных операций чтения, то он переводится в состояние ожидания. Если количество инициализированных и не завершенных операций чтения меньше s_r, то в очередь диска D помещается пакет E с *адресом получателя* $\alpha(E)=P$ и *адресом отправителя* $\varsigma(E)=D$.

Операция записи. Пусть процессорному модулю P требуется записать пакет E на диск $D\in\mathfrak{D}$. Если процессор P ранее инициализировал s_w незавершенных операций записи, то он переводится в состояние ожидания. Если количество инициализированных и не завершенных операций записи меньше s_w, то в очередь сетевого концентратора $N=F(P)$ помещается пакет E с адресом получателя $\alpha(E)=D$ и адресом отправителя $\varsigma(E)=P$

Модуль сетевого концентратора $N\in\mathfrak{N}$ осуществляет пассивную передачу пакетов по соединительной сети, перманентно выполняя следующую процедуру:

```
procedure // Монитор модуля сетевого концентратора N
    извлечь пакет E из очереди N;
    if α(E)∉T(N) then
        поместить E в очередь F(N);
    else
        найти M∈𝔐 такой, что F(M)=N & α(E)∈T(M);
        if α(E)=M & M∈𝔓 then
            уменьшить на 1 количество незавершенных чтений, инициированных M;
            if M находится в состоянии ожидания по ограничению s_r then
                перевести M в активное состояние;
            end if;
        else // M∈𝔇∪𝔑
            поместить E в очередь M;
        end if;
    end if;
end procedure.
```

Дисковый модуль $D\in\mathfrak{D}$ осуществляет пассивные чтение и запись пакетов, перманентно выполняя следующую процедуру:

```
procedure // Монитор дискового модуля D
   извлечь пакет E из очереди D;
   if α(E)=D then
      P:=ς(E); // отправитель
      уменьшить на 1 количество незавершенных операций записи у P;
      if P находится в состоянии ожидания по ограничению s_w then
         перевести P в активное состояние;
      end if;
   else // α(E)∈𝔓
      поместить E в очередь F(D);
   end if;
end procedure.
```

В модели *DMM* процесс обработки данных организуется в виде цикла, выполняющего стандартную последовательность шагов, называемую *тактом*. Такт определяется как следующая последовательность действий:

1) каждый модуль сетевого концентратора обрабатывает все пакеты, ожидающие передачи;
2) каждый активный процессорный модуль выполняет одну операцию чтения или записи;
3) каждый дисковый модуль обрабатывает один пакет из своей очереди.

Очевидно, что в этом случае в очереди любого концентратора не может одновременно находиться более $2|\mathfrak{P}|$ пакетов, в очереди любого диска не может находиться более $s_r s_w|\mathfrak{P}|$ пакетов.

Стоимостная модель

С каждым модулем $M\in\mathfrak{M}$ связывается *коэффициент трудоемкости* $h_M \in \mathbb{R}, \quad 1 \le h_M < +\infty$. Так как время обработки процессором одного пакета для OLTP-приложений приблизительно в 10^5-10^6 раза быстрее, чем время обмена с диском или передачи по сети, то мы полагаем

$$h_P = 1, \quad \forall P \in \mathfrak{P}.$$

Так как модуль сетевого концентратора за один такт может передавать несколько пакетов, то для каждого модуля сетевого концентратора $N\in\mathfrak{N}$ мы вводим функцию помех

$$f_N(m_i^N) = e^{\frac{m_i^N}{\delta_N}}.$$

Здесь m_i^N обозначает число пакетов, проходящих через N на i-том такте; $\delta_N>1$ – масштабирующий коэффициент. Таким образом, время, требуемое модулю сетевого концентратора N для выполнения i-того такта, вычисляется по формуле

$$t_i^N = h_N f_N(m_i^N), \qquad \forall N \in \mathfrak{N}.$$

Положим $H_{\mathfrak{D}} = \max_{D\in\mathfrak{D}}(h_D)$. Тогда общее время работы системы, затраченное на обработку смеси транзакций в течении k тактов, вычисляется по формуле

$$T = \sum_{i=1}^{k} \max(\max_{N\in\mathfrak{N}}(t_i^N), H_{\mathfrak{D}})$$

Заключение

На основе предложенной модели нами был спроектирован и реализован программный комплекс, получивший название ЭВМБД (Эмулятор Виртуальных Мультипроцессоров Баз Данных). Данный программный комплекс может быть использован для моделирования и сравнительного анализа различных иерархических многопроцессорных архитектур в контексте задач класса OLTP. Был проведен тестовый эксперимент по моделированию архитектуры высокопроизводительного вычислительного кластера Южно-Уральского государственного университета [http://cluster.susu.ru/]. Полученные на эмуляторе результаты достаточно хорошо согласуются с реальными результатами, полученными при использовании прототипа параллельной СУБД Омега [http://omega.susu.ru/].

ЛИТЕРАТУРА

1. *Соколинский Л.Б.* Обзор архитектур параллельных систем баз данных // Программирование. -2004. -№6. -С. 49-63.
2. *Foster I.T., Grossman R.L.* Blueprint for the future of high-performance networking: Data integration in a bandwidth-rich world // Communications of the ACM. -2003. -Vol. 46, No. 11. -P. 50-57.

3. *Bhide A., Stonebraker M.* A Performance Comparison of Two Architectures for Fast Transaction Processing // Proceedings of the Fourth International Conference on Data Engineering, February 1-5, 1988, Los Angeles, California, USA. IEEE Computer Society. -1988. -P. 536-545.
4. *Bhide A.* An Analysis of Three Transaction Processing Architectures // Fourteenth International Conference on Very Large Data Bases (VLDB'88), August 29 - September 1, 1988, Los Angeles, California, USA, Proceedings. Morgan Kaufmann. -1988. -P. 339-350.
5. *Bouganim L., Florescu D., Valduriez P.* Dynamic Load Balancing in Hierarchical Parallel Database Systems // VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India. Morgan Kaufmann. -1996. -P. 436-447.
6. *Xu Y., Dandamudi S.P.* Performance Evaluation of a Two-Level Hierarchical Parallel Database System // Proceedings of the Int. Conf. Computers and Their Applications, Tempe, Arizona. -1997. -P. 242-247.