На правах рукописи

ЛЕПИХОВ Андрей Валерьевич

МЕТОДЫ ОБРАБОТКИ ЗАПРОСОВ В СИСТЕМАХ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ ДЛЯ МНОГОПРОЦЕССОРНЫХ СИСТЕМ С ИЕРАРХИЧЕСКОЙ АРХИТЕКТУРОЙ

05.13.11 - математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата физико-математических наук

Москва – 2008

Работа выполнена на кафедре системного программирования
Южно-Уральского государственного университета.

Научный руководитель: доктор физико-математических наук, профессор
СОКОЛИНСКИЙ Леонид Борисович.

Официальные оппоненты: доктор технических наук, профессор
КУЗНЕЦОВ Сергей Дмитриевич;

кандидат физико-математических наук
ЖУМАТИЙ Сергей Анатольевич.

Ведущая организация: Санкт-Петербургский
государственный университет.

Защита состоится 19 декабря 2008 года в 15 часов на заседании диссертационного совета Д 501.002.09 при Московском государственном университете им. М.В. Ломоносова по адресу: 119992, г. Москва, Ленинские горы, д. 1, стр. 4, НИВЦ МГУ, конференц–зал.

С диссертацией можно ознакомиться в библиотеке НИВЦ МГУ.

Автореферат разослан 17 ноября 2008 года.

Ученый секретарь
диссертационного совета  Суворов В.В.

Общая характеристика работы

Актуальность темы. В настоящее время все большее распространение получают параллельные системы баз данных, ориентированные на мультипроцессоры с иерархической архитектурой. Это связано с тем, что современные многопроцессорные системы в большинстве случаев организуются по иерархическому принципу. 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. Если в системе используются еще и многоядерные процессоры, то мы получаем третий уровень иерархии. Другим источником многопроцессорных иерархий являются грид-технологии, позволяющие объединять несколько различных суперкомпьютеров в единую вычислительную систему. Подобная грид-система будет иметь многоуровневую иерархическую структуру. На нижних уровнях иерархии располагаются процессоры отдельных кластерных систем, соединенные высокоскоростной внутренней сетью. На верхних уровнях располагаются вычислительные системы, объединенные корпоративной сетью. Высший уровень иерархии может представлять сеть Интернет.

Иерархические многопроцессорные системы обладают рядом особенностей. В рамках одного уровня иерархии скорость обмена сообщениями между вычислительными узлами практически одинакова. При переходе на более высокие уровни иерархии скорость межпроцессорных обменов может уменьшаться на несколько порядков. Следовательно, системы управления базами данных (СУБД) для многопроцессорных иерархий, или кратко – *иерархические СУБД*, занимают промежуточное положение между параллельными СУБД, для которых характерна одинаковая скорость межпроцессорных обменов, и распределенными СУБД, в которых скорость обмена сообщениями может значительно отличаться для разных пар вычислительных узлов. Анализ показывает, что не все алгоритмы и методы, используемые в параллельных и распределенных СУБД, могут быть перенесены в иерархические СУБД, а те алгоритмы и методы, которые такой перенос допускают, как правило, нуждаются в существенной адаптации. В соответствии с этим *актуальной* является задача разработки эффективных алгоритмов и методов обработки запросов, ориентированных на применение в системах баз данных с иерархической многопроцессорной архитектурой, масштабируемой до десятков тысяч процессорных узлов.

Цель и задачи исследования. *Целью* диссертационного исследования является разработка эффективных методов и алгоритмов обработки запросов, размещения данных и балансировки загрузки, ориентированных на многопроцессорные системы с иерархической архитектурой и их реализация в прототипе

иерархической СУБД. Для достижения этой цели необходимо было решить следующие ***задачи***:

1. Разработать и аналитически исследовать стратегию размещения и репликации базы данных для многопроцессорных иерархических систем.
2. Разработать эффективный алгоритм динамической балансировки загрузки на основе предложенной стратегии размещения данных.
3. Разработать метод параллельной обработки запросов для многопроцессорных иерархий, использующий предложенные стратегию размещения данных и алгоритм балансировки загрузки.
4. Реализовать разработанные методы и алгоритмы в прототипе иерархической СУБД «Омега».
5. Провести вычислительные эксперименты для оценки эффективности предложенных решений.

Методы исследования. Проведенные в работе исследования базируются на реляционной модели данных и используют методы системного программирования. Для решения поставленных задач применялся математический аппарат, в основе которого лежит теория графов, предоставляющая возможность изучения и моделирования многопроцессорных иерархических конфигураций систем баз данных.

Научная новизна работы заключается в следующем:

1) предложена модель симметричной многопроцессорной иерархической системы;
2) предложен метод частичного зеркалирования, использующий функцию репликации, которая сопоставляет каждому уровню иерархии определенный коэффициент репликации;
3) получены аналитические оценки трудоемкости формирования и обновления реплик в методе частичного зеркалирования;
4) разработан новый алгоритм балансировки загрузки для параллельных СУБД с иерархической архитектурой;
5) разработан метод обработки запросов, допускающий эффективную балансировку загрузки для иерархических систем баз данных.

Теоретическая ценность состоит в том, что дано формальное описание симметричной многопроцессорной системы с иерархической архитектурой. Представлены доказательства теорем об оценке размера реплик и оценке трудоемкости формирования реплик без учета помех для метода частичного зеркалирования. **Практическая ценность** работы заключается в том, что предложенный метод частичного зеркалирования совместно с разработанным алгоритмом балансировки загрузки может использоваться для решения проблемы перекосов по данным в широком классе приложений систем баз данных для вычислительных кластеров и грид-систем.

Апробация работы. Основные положения диссертационной работы, разработанные модели, методы, алгоритмы и результаты вычислительных экспериментов докладывались автором на следующих международных и всероссийских научных конференциях:

- на Четвертом весеннем коллоквиуме молодых исследователей в области баз данных и информационных систем (SYRCoDIS) (1–2 июня 2006 г., Москва);
- на Всероссийской научной конференции «Научный сервис в сети Интернет: технологии параллельного программирования» (18-23 сентября 2006 г., Новороссийск);
- на Всероссийской научной конференции «Научный сервис в сети Интернет: решение больших задач» (22-27 сентября 2008 г., Новороссийск);
- на Международной научной конференции «Параллельные вычислительные технологии» (29 января – 2 февраля 2007 г., Челябинск).

Публикации. Основные научные результаты диссертации опубликованы в 6 печатных работах, приведенных в конце автореферата. Статья [1] опубликована в научном журнале «Автоматика и телемеханика», включенном ВАК в перечень журналов, в которых должны быть опубликованы основные результаты диссертаций на соискание ученой степени доктора наук. В статье [1] А.В. Лепихову принадлежит раздел 3 (стр. 118-124). В работах [4, 5] Л.Б. Соколинскому принадлежит постановка задачи; А.В. Лепихову принадлежат все полученные результаты.

Структура и объем работы. Диссертация состоит из введения, четырех глав, заключения и библиографии. Объем диссертации составляет 102 страницы, объем библиографии – 113 наименований.

Содержание работы

Во введении приводится обоснование актуальности темы, формулируются цели работы, ее новизна и практическая значимость; приводится обзор работ по тематике исследования и кратко излагается содержание диссертации.

Первая глава, «Многопроцессорные иерархии», посвящена исследованию многопроцессорных вычислительных систем с иерархической архитектурой.

Иерархические многопроцессорные системы состоят из нескольких уровней иерархии. Первый уровень иерархии представлен многоядерными процессорами. На втором уровне иерархии многоядерные процессоры объединяются в процессорные узлы с SMP-архитектурой. На третьем уровне иерархии процессорные узлы, объединяются в кластер. На четвертом уровне кластеры объединяются в кооперативную грид-систему. На пятом уровне кооперативные грид-системы объединяются в ассоциацию, образующую

корпоративную грид-систему. Шестой уровень представлен глобальными грид-сетями, которые используют в качестве коммуникационной сети сеть Интернет. Дадим следующее определение многопроцессорной иерархии.

Иерархическая многопроцессорная система – это многопроцессорная система, в которой процессоры объединяются в единую систему с помощью соединительной сети, имеющей иерархическую структуру и обладающую свойствами однородности по горизонтали и неоднородности по вертикали. *Однородность по горизонтали* означает, что в пределах одного уровня иерархии скорость обменов между двумя процессорами является постоянной величиной, независимо от того в каком поддереве иерархии эти процессоры находятся. *Неоднородность по вертикали* означает, что скорость обменов на разных уровнях иерархии существенно различается и этот факт должен учитываться в алгоритмах обработки запросов.

Далее строится симметричная модель многопроцессорной иерархии применительно к системам баз данных. В основе симметричной модели лежит понятие DM-дерева, являющегося ориентированным деревом, узлы которого относятся к одному из трех классов:

$\mathfrak{P}(T)$ – класс «процессорные модули»;

$\mathfrak{D}(T)$ – класс «дисковые модули»;

$\mathfrak{N}(T)$ – класс «модули сетевых концентраторов».

С каждым узлом $v \in \mathfrak{M}(T)$ в *DM*-дереве T связывается *коэффициент трудоемкости* $\eta(v)$, являющийся вещественным числом, большим либо равным единицы. Коэффициент трудоемкости определяет время, необходимое узлу для обработки некоторой порции данных.

DM-деревья A и B называются *изоморфными*, если существуют взаимно однозначное отображение f множества $\mathfrak{M}(A)$ на множество $\mathfrak{M}(B)$ и взаимно однозначное отображение g множества $\mathfrak{E}(A)$ на множество $\mathfrak{E}(B)$ такие, что:

1) узел v является конечным узлом дуги e в дереве A тогда и только тогда, когда узел $\mathrm{f}(v)$ является конечным узлом дуги $\mathrm{g}(e)$ в дереве B;

2) узел w является начальным узлом дуги e в дереве A тогда и только тогда, когда узел $\mathrm{f}(w)$ является начальным узлом дуги $\mathrm{g}(e)$ в дереве B;

3) $p \in \mathfrak{P}(A) \quad \Leftrightarrow \quad \mathrm{f}(p) \in \mathfrak{P}(B)$;

4) $d \in \mathfrak{D}(A) \quad \Leftrightarrow \quad \mathrm{f}(d) \in \mathfrak{D}(B)$;

5) $n \in \mathfrak{N}(A) \quad \Leftrightarrow \quad \mathrm{f}(n) \in \mathfrak{N}(B)$;

6) $\eta(\mathrm{f}(v)) = \eta(v)$.

Упорядоченная пара отображений $\mathrm{q} = (\mathrm{f}, \mathrm{g})$ называется *изоморфизмом* *DM*-дерева A на *DM*-дерево B.

DM-дерево *T* высоты *H* называется *симметричным*, если выполняются следующие условия:

1) любые два смежных поддерева уровня $l<H$ являются изоморфными;
2) любое поддерево уровня H-1 содержит в точности один диск и один процессор.

Далее проводится анализ параллельной, распределенной и иерархической СУБД, который показывает, что иерархическая СУБД совмещает в себе черты параллельной и распределенной СУБД. Эта двойственная природа иерархической СУБД вытекает из базовых свойств многопроцессорной иерархии: однородность по горизонтали и неоднородность по вертикали. Неоднородность по вертикали сближает иерархические СУБД с распределенными, а однородность по горизонтали – с параллельными.

Далее описываются две модели реализации параллельных исполнителей запросов, основанные на использовании скобочного шаблона и оператора обмена **exchange**.

Скобочный шаблон используется для унифицированного представления узлов дерева запроса. В качестве основных методов здесь фигурируют функции **reset** и **next**, реализующие итератор. После оптимизации запроса СУБД «вставляет» в каждый скобочный шаблон ту или иную реализацию соответствующей реляционной операции. Связь скобочного шаблона с конкретной реализацией операции осуществляется путем добавления еще одного специального атрибута в скобочный шаблон – «указатель на функцию реализации операции».

Оператор обмена **exchange** реализует концепцию операторной модели. Отличительной особенностью оператора **exchange** является то, что он может быть помещен в любом месте дерева запроса, не оказывая при этом никакого влияния на соседние операторы. Он инкапсулирует в себе все механизмы, необходимые для реализации фрагментного параллелизма и обеспечивает прозрачную реализацию параллельных алгоритмов.

В заключении главы описан механизм параллельной обработки запросов на примере двухуровневой иерархической архитектуры. Описывается последовательность формирования параллельного плана и механизм параллельной обработки запроса в виде набора *параллельных агентов*, каждый из которых обрабатывает отдельный фрагмент отношения на выделенном ему процессорном узле. Полученные агентами результаты *сливаются* в результирующее отношение.

Во второй главе, «Размещение данных и балансировка загрузки» предлагается новый подход к размещению данных и балансировке загрузки в многопроцессорных иерархиях. В основе предлагаемого подхода лежат стратегии распределения и репликации базы данных. Математической базой данного подхода является симметричная модель многопроцессорной иерархической системы.

Стратегия размещения данных предполагает использование горизонтальной фрагментации отношений. Каждое отношение разбивается на непересекающиеся горизонтальные фрагменты, которые размещаются на различных дисках. Фрагмент делится на логические сегменты, между которыми определено отношение порядка, называемое естественным порядком. На практике естественный порядок может определяться физическим порядком следования кортежей или индексом. Будем обозначать количество кортежей через $\mathrm{T}(F)$, а длину сегмента через $\mathrm{L}(F)$.

Стратегия репликации описывается следующим образом. На каждом дисковом модуле $d_i \in \mathfrak{D}(T) \quad (i > 0)$ фрагмент может иметь несколько (возможно неполных) зеркальных копий, называемых репликами, которые располагаются на других дисках. Размер реплики F_i задается *коэффициентом репликации* $\rho_i \in \mathbb{R}, \quad 0 \le \rho_i \le 1$, являющимся атрибутом реплики F_i и вычисляется по следующей формуле

$$\mathrm{T}(F_i) = \mathrm{T}(F_0) - \left\lceil (1 - \rho_i) \cdot \mathrm{S}(F_0) \right\rceil \cdot \mathrm{L}(F_0). \qquad (1)$$

Пусть фрагмент F_0 располагается на диске $d_0 \in \mathfrak{D}(T)$. Мы будем использовать следующий метод для построения реплики F_i на диске $d_i \in \mathfrak{D}(T) \quad (i > 0)$, называемый *методом частичного зеркалирования*. Построим последовательность поддеревьев дерева T

$$\{M_0, M_1, \ldots, M_{H-2}\}, \qquad (2)$$

обладающую следующими свойствами:

$$\begin{cases} l(M_j) = j \\ d_0 \in \mathfrak{D}(M_j) \end{cases} \qquad (3)$$

для всех $0 \le j \le H - 2$. Здесь $l(M_j)$ обозначает уровень поддерева M_j. Для любого симметричного дерева T существует только одна такая последовательность. Будем считать, что

$$\rho_i = r(j). \qquad (4)$$

Для формирования реплики F_i на диске d_i используется описанная выше стратегия репликации с коэффициентом репликации, определяемым по формуле (4).

Следующая теорема дает оценку для размера реплики в методе частичного зеркалирования.

Теорема 1. Пусть T – симметричное *DM*-дерево высоты $H = h(T) > 0$. Пусть фрагмент F_0 располагается на диске $d_0 \in \mathfrak{D}(T)$. Пусть M поддерево дерева T такое, что $1 \le l(M) \le H - 1$ и $d_0 \in \mathfrak{D}(M)$. Пусть M' – произвольное смежное с M поддерево дерева T. Тогда для любого $d_i \in \mathfrak{D}(M')$ справедлива следующая оценка для размера реплики F_i фрагмента F_0, размещенной на диске d_i:

$$\mathrm{T}(F_i) = r\big(l(M)-1\big)\mathrm{T}(F_0) + \mathrm{O}\big(\mathrm{L}(F_0)\big),$$

где $\mathrm{L}(F_0)$ – длина сегмента для фрагмента F_0.

Оценка суммарного размера всех реплик фрагмента может быть получена с помощью следующей теоремы.

Теорема 2. Пусть T – симметричное *DM*-дерево высоты $H \ge 2$. Пусть фрагмент F_0 располагается на диске $d_0 \in \mathfrak{D}(T)$. Обозначим степень уровня l дерева T как δ_l. Обозначим $\mathrm{R}(F_0) = \sum_{i=1}^{|\mathfrak{D}(T)|} \mathrm{T}(F_i)$ – суммарное количество кортежей во всех репликах фрагмента F_0. Тогда

$$\mathrm{R}(F_0) = \mathrm{T}(F_0)\sum_{j=0}^{H-2} r(j)(\delta_j - 1)\prod_{k=j+1}^{H-2}\delta_k + \mathrm{O}\big(\mathrm{L}(F_0)\big). \qquad (5)$$

Далее проводится анализ функции репликации. При выборе функции репликации $r(l)$ целесообразно учитывать коэффициенты трудоемкости узлов *DM*-дерева. Очевидно, что в симметричном *DM*-дереве все вершины уровня l имеют одинаковую трудоемкость $\eta(l)$. Назовем симметричное *DM*-дерево T *регулярным*, если для любых двух уровней l и l' дерева T справедливо

$$l < l' \quad \Rightarrow \quad \eta(l) \ge \eta(l'), \qquad (6)$$

Следующая теорема позволяет получить оценку трудоемкости покортежного формирования реплики в регулярном *DM*-дереве.

Теорема 3. Пусть T – регулярное *DM*-дерево высоты $H > 0$. Пусть фрагмент F_0 располагается на диске $d_0 \in \mathfrak{D}(T)$. Пусть M поддерево дерева T такое, что $1 \le l(M) \le H-2$ и $d_0 \in \mathfrak{D}(M)$. Пусть M' – произвольное смежное с M поддерево дерева T; F_i – реплика фрагмента F_0, размещенная на диске $d_i \in \mathfrak{D}(M')$. Обозначим $\tau(F_i)$ – трудоемкость покортежного формирования реплики F_i при отсутствии помех. Тогда

$$\tau(F_i) = \eta\big(l(M)-1\big)\cdot r\big(l(M)-1\big)\mathrm{T}\big(F_0\big) + \mathrm{O}\big(\eta_0\big),$$

где $\eta_0 = \eta(0)$ – коэффициент трудоемкости корня дерева T.

Оценка трудоемкости покортежного формирования всех реплик фрагмента без учета помех может быть получена с помощью следующей теоремы.

Теорема 4. Пусть T – регулярное *DM*-дерево высоты $H \ge 2$. Пусть фрагмент F_0 располагается на диске $d_0 \in \mathfrak{D}(T)$. Обозначим степень уровня l дерева T как δ_l. Обозначим $\tau(F_0) = \sum_{i=1}^{|\mathfrak{D}(T)|} \tau(F_i)$ – суммарная трудоемкость покортежного формирования всех реплик фрагмента F_0 без учета помех. Тогда

$$\tau(F_0) = \mathrm{T}\big(F_0\big)\sum_{j=0}^{H-2}\eta\big(j\big)r\big(j\big)\big(\delta_j - 1\big)\prod_{k=j+1}^{H-2}\delta_k + \mathrm{O}\big(\eta_0\big). \qquad (7)$$

Определим рекурсивно *нормальную* функцию репликации $r(l)$ следующим образом:

1) для $l = H-2$: $r(H-2) = \dfrac{1}{\eta(H-2)(\delta_{H-2}-1)}$;

2) для $0 \le l < H-2$: $r(l) = \dfrac{\eta(l+1)(\delta_{l+1}-1)r(l+1)}{\eta(l)(\delta_l-1)\delta_{l+1}}$.

Справедлива следующая теорема.

Теорема 5. Пусть T – регулярное *DM*-дерево высоты $H \ge 2$. Пусть $\mathbb{F}$ – множество фрагментов, составляющих базу данных. Пусть $\mathbb{R}$ – множество всех реплик всех фрагментов из множества $\mathbb{F}$, построенных с использованием нормальной функции репликации. Пусть $\mathrm{T}(\mathbb{F})$ – размер базы данных в кортежах (здесь мы предполагаем, что все кортежи имеют одинаковую длину в байтах), $\tau(\mathbb{R})$ – суммарная трудоемкость покортежного формирования всех реплик без учета помех. Тогда

$$\tau(\mathbb{R}) \approx k\,\mathrm{T}(\mathbb{F}),$$

где k – некоторая константа, не зависящая от $\mathbb{F}$.

Данная теорема показывает, что при использовании нормальной функции репликации трудоемкость обновления реплик в регулярной многопроцессорной иерархической системе пропорциональна размеру обновляемой части базы данных при условии, что соединительная сеть обладает достаточной пропускной способностью.

Далее описывается предлагаемый в диссертационном исследовании метод балансировки загрузки. Пусть задан некоторый запрос $\mathbb{Q}$, имеющий n входных отношений. Пусть $\mathfrak{Q}$ – параллельный план запроса $\mathbb{Q}$. Каждый агент $Q \in \mathfrak{Q}$ имеет n входных потоков $s_1, \ldots, s_n$. Входной поток представляет собой структуру, абстрагирующую параллельного агента от входного отношения.

На начальном этапе выполнения запроса выполняется инициализация агента, Затем агент переводится в активное состояние и начинает обработку данных, поставляемых его входными потоками. После того, как все данные из потока выработаны, агент переходит в пассивное состояние.

При выполнении параллельного плана запроса одни агенты могут завершить свою работу и находиться в пассивном состоянии в то время, как другие агенты будут продолжать обработку назначенных им отрезков. Для таких случаев предлагается алгоритм балансировки загрузки. Суть данного алгоритма заключается в передаче части необработанных сегментов от перегруженного агента (агента-лидера) простаивающему агенту (агенту-аутсайдеру).

Далее предлагается оптимистическая стратегия выбора аутсайдера и функция балансировки. *Оптимистическая стратегия* основана на механизме рейтингов. Каждому агенту параллельного плана в процессе балансиров-

ки загрузки присваивается рейтинг. В качестве аутсайдера всегда выбирается агент, имеющий максимальный положительный рейтинг. Если таковые агенты отсутствуют, то освободившийся агент завершает свою работу. Для вычисления рейтинга оптимистическая стратегия использует *рейтинговую функцию* $\gamma: \mathfrak{Q} \to \mathbb{R}$ следующего вида:

$$\gamma\left(\tilde{Q}\right) = \tilde{a}_i \operatorname{sgn}\left(\max_{1 \le i \le n}\left(\tilde{q}_i\right) - B\right) \lambda r\left(l(\tilde{\tilde{M}})\right) \sum_{i=1}^{n} \tilde{q}_i \,.$$

Здесь λ – положительный весовой коэффициент, регулирующий влияние коэффициента репликации на величину рейтинга; B – целое неотрицательное число, задающее нижнюю границу количества сегментов, которое можно передавать при балансировке нагрузки.

Функция балансировки Δ для каждого потока $\tilde{s}_i$ агента-аутсайдера определяет количество сегментов, передаваемых агенту-лидеру на обработку. Мы предлагаем следующую функцию балансировки:

$$\Delta(\tilde{s}_i) = \min\left(\lceil \tilde{q}_i / 2 \rceil, r\left(l(\tilde{\tilde{M}})\right) \mathrm{S}\left(\tilde{f}_i\right)\right).$$

где q_i – количество сегментов в отрезке, подлежащем обработке.

В третьей главе, «Иерархическая СУБД «Омега», описывается реализация прототипа иерархической СУБД «Омега» для многопроцессорных иерархий. СУБД «Омега» включает в себя следующие подсистемы*: Пользователь, Клиент, Координатор* и *Ядро СУБД.*

Клиент представляет собой подсистему иерархической СУБД, реализующую интерфейс пользователя. Для каждого пользователя иерархической СУБД порождается отдельный экземпляр Клиента. Клиент допускает вызов в формате командной строки и может быть использован программными системами, являющимися внешними по отношению к данной СУБД. Поступающие от пользователя запросы Клиент передает Координатору.

Основной задачей *Координатора* является доставка запроса от Клиента на Ядра СУБД. Координатор принимает запрос от каждого обратившегося к нему Клиента и передает его Ядрам СУБД, которые будут выполнять обработку данного запроса.

Ядро СУБД непосредственно выполняет обработку запроса и передает результат тому Координатору, от которого поступил запрос.

Варианты использования подсистемы «Клиент» проектируются таким образом, чтобы обеспечить диалоговый и пакетный режимы работы. Основой проектирования вариантов использования подсистемы «Координатор» является уменьшение времени отклика на запрос. Для организации приема запросов от пользователей в данной СУБД использован реляционный язык запросов RQL. Результатом обработки запроса является результирующая таблица и лог-файл.

Далее описывается реализация оператора обмена **`exchange`**, который обеспечивает параллельное выполнение запроса. Оператор **`exchange`** вводит в набор физических операций четыре новых оператора: **`merge`**, **`split`**,

scatter и **gather**, каждый из которых реализован в соответствии с итераторной моделью управления.

Оператор **split** определяет стратегию расщепления кортежей, поступающих из входного потока на «свои» и «чужие». В данной реализации оператор **split** обрабатывает кортежи по одному. При этом обработка очередного кортежа начинается только в том случае, если оператор **scatter** готов к приему кортежа.

Операторы **scatter** и **gather** определяют стратегию обменов кортежами между параллельными агентами в процессе обработки запроса. В данной реализации предлагается стратегия пакетированной передачи кортежей, предназначенная для эффективного использования коммуникационной сети.

Оператор **merge** определяет стратегию чтения кортежей с диска, и из сети. В данной реализации предлагается схема попеременного опроса левого и правого сына. Данная схема позволяет устранить дисбаланс между операциями обмена с сетью и диском. В настоящей реализации операция чтения кортежей из сети признается равноправной операции чтения кортежей с диска.

Далее описывается реализация *механизма балансировки загрузки*. Реализация механизма балансировки загрузки в иерархической СУБД «Омега» обеспечивается подсистемами *Менеджер параллельных агентов* и *Менеджер заданий*. Менеджер параллельных агентов выполняет функции управления параллельными агентами. Менеджер заданий выполняет функции сервера статистики, отражающий ход обработки запроса.

В предлагаемой реализации процедура балансировки загрузки носит асинхронный характер. При возникновении ситуации балансировки интерпретатор передает менеджеру агентов сообщение о необходимости выполнения балансировки и не прерывает обработку запроса.

В основе данной реализации механизма балансировки лежит понятие кванта времени, который определяет периодичность, с которой параллельный агент должен отправлять отчет о состоянии входных потоков менеджеру заданий. Такая схема позволяет свести к минимуму накладные расходы в случае неуспешной балансировки загрузки и уменьшить время простоя параллельного агента.

В четвертой главе, «Вычислительные эксперименты», приводится описание экспериментов, проведенных для проверки эффективности предложенных методов и алгоритмов. Для исследования алгоритма балансировки загрузки в прототипе иерархической СУБД «Омега» был использован алгоритм соединения хешированием в оперативной памяти. При проведении экспериментов использовалась тестовая база данных, состоящая из отношений с целочисленными атрибутами. Для формирования значений атрибута фрагментации использовалась вероятностная модель, в соответствии с которой размер каждого фрагмента отношения определяется *коэффициентом перекоса* θ. Для формирования значений атрибута соединения использовался

коэффициент перекоса μ, в соответствии с которым кортежи подразделяются на два класса: «*свои*» и «*чужие*» Коэффициент μ указывает процентное содержание «чужих» кортежей во фрагменте.

В первой серии экспериментов были исследованы параметры балансировки загрузки, такие как интервал балансировки и размер сегмента. Исследование интервала балансировки загрузки (см. рис. 1) показывает, что накладные расходы при увеличении частоты опроса узлов не оказывают существенного воздействия на итоговое время выполнения запроса. Этот факт достигается благодаря асинхронному методу реализации балансировки загрузки в прототипе иерархической СУБД «Омега». Проведенные эксперименты показывают, что хорошим выбором будет использование интервала балансировки равного 1 секунде.

Исследование влияния размера сегмента на эффективность балансировки (см. рис. 2) показывает, что сегменты небольшого размера ухудшают показатели балансировки, поскольку существенно возрастает количество «мелких» балансировок, что ведет к росту накладных расходов на перераспределение заданий между параллельными агентами. Большие значения размера сегмента также являются неэффективными, так как в этом случае размер сегмента становится сравнимым с размером фрагмента, что делает балансировку невозможной. Результаты экспериментов показывают, что оптимальным будет промежуточное значение, которое в данном случае равно 20 000 кортежей. Эта величина составляет примерно 0.01% от средней величины фрагмента.

Во второй серии экспериментов исследовалось влияние метода балансировки загрузки на время обработки запросов. Результаты данных испытаний показаны на рис. 3. Данные эксперименты показывают, что оптимальным выбором является значение $\rho=0.8$, которое позволяет практически полностью устранить негативное влияние перекосов по атрибуту фрагментации.

Последняя серия экспериментов была посвящена исследованию масштабируемости предложенного в диссертационной работе метода балансировки загрузки. Результаты этих экспериментов представлены на рис. 4, рис. 5 и рис. 6. В экспериментах, показанных на графике 4, было исследовано влияние величины коэффициента перекоса по атрибуту соединения на ускорение. Результаты данных экспериментов показывают, что балансировка загрузки дает наибольший эффект при значении $\mu=50\%$.

В экспериментах, показанных на графике 5, исследовалось влияние коэффициента репликации на коэффициент ускорения. Результаты данных экспериментов демонстрируют увеличение коэффициента ускорения при выполнении балансировки загрузки. При этом следует отметить, что в данном случае хорошим выбором будет $\rho=0.8$.

В экспериментах, показанных на графике 6, исследовалось влияние перекоса по атрибуту фрагментации на коэффициент ускорения при выполнении балансировки загрузки. Несмотря на то, что перекосы существенно ухудшают значение коэффициента ускорения, балансировка загрузки обеспечивает ощутимое ускорение даже при больших значениях коэффициента перекоса.

В заключении суммируются основные результаты диссертационной работы, выносимые на защиту, приводятся данные о публикациях и апробациях автора по теме диссертации, и рассматриваются направления дальнейших исследований в данной области.

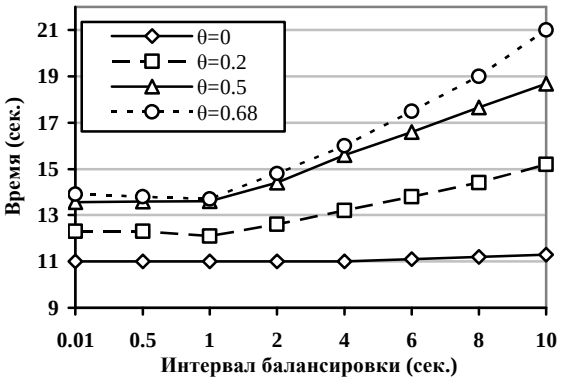

Рис 1. Зависимость времени выполнения запроса от интервала балансировки (n=64, μ =50%).

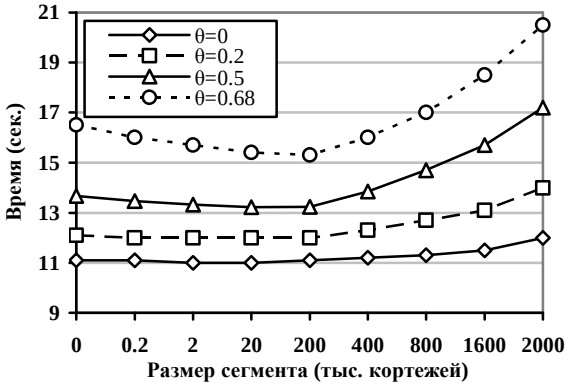

Рис 2. Зависимость времени выполнения запроса от размера сегмента (n=64, μ =50%).

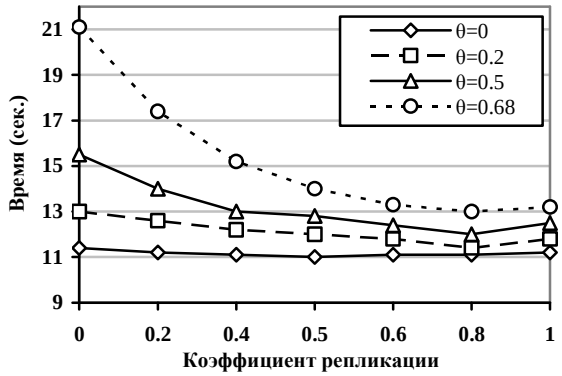

Рис 3. Влияние коэффициента репликации на время выполнения запроса (μ =50%, n=64).

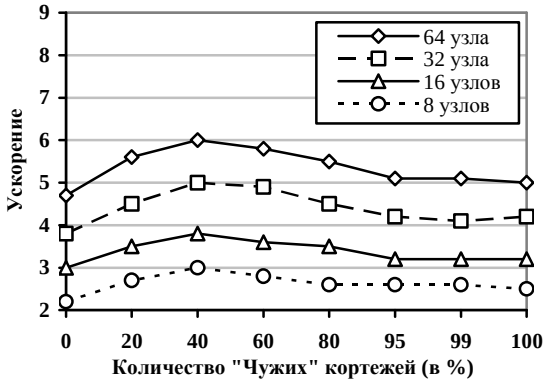

Рис 4. Зависимость коэффициента ускорения от перекоса по значению атрибута соединения μ (n=64, ρ=0.5).

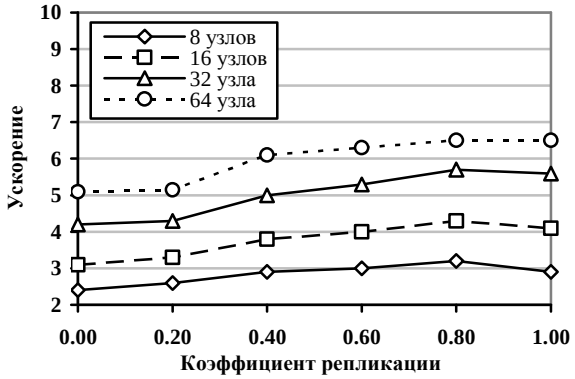

Рис 5. Зависимость ускорения от коэффициента репликации ρ. (n=64, μ=50%).

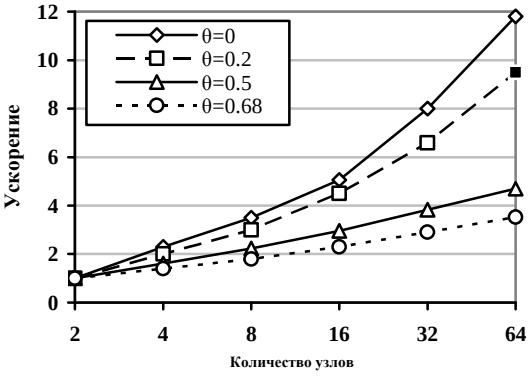

Рис 6. Влияние перекоса по данным на коэффициент ускорения (μ=50%, ρ=0.50).

Основные результаты диссертационной работы

На защиту выносятся следующие новые научные результаты.

1. Построена математическая модель многопроцессорной иерархии. На основе этой модели разработан метод частичного зеркалирования, который может быть использован для динамической балансировки загрузки. Доказаны теоремы, позволяющие получить аналитическую оценку трудоемкости формирования и обновления реплик при использовании метода частичного зеркалирования.
2. Предложен метод параллельной обработки запросов в иерархических многопроцессорных системах, позволяющий осуществлять эффективную динамическую балансировку загрузки на базе техники частичного зеркалирования.
3. Разработан прототип иерархической СУБД «Омега», реализующий предложенные методы и алгоритмы. Проведены тестовые испытания СУБД «Омега» на вычислительных кластерах, входящих в грид-систему «СКИФ-Полигон», подтвердившие эффективность предложенных алгоритмов, методов и подходов.

Публикации по теме диссертации

Статьи, опубликованные в научных журналах из списка ВАК

1. *Лепихов А.В.*Технологии параллельных систем баз данных для иерархических многопроцессорных сред / *Лепихов А.В., Соколинский Л.Б., Костенецкий П.С.* // Автоматика и телемеханика. –2007. –No. 5. –С. 112-125.

Другие публикации

2. *Лепихов А.В.* Модель вариантов использования параллельной системы управления базами данных для грид // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». –Челябинск : ЮУрГУ, 2008 г. –No. 15 (115). –Вып. 1. –С. 42–53.
3. *Лепихов А.В.* Балансировка загрузки при выполнении операций соединения в параллельных СУБД для кластерных систем // Научный сервис в сети Интернет: решение больших задач. Труды Всероссийской научной конференции (22–27 сентября 2008 г., г. Новороссийск). –М.: Изд-во МГУ, 2008. –С. 292–295.
4. *Лепихов А.В.* Стратегия размещения данных в многопроцессорных системах с симметричной иерархической архитектурой / *А.В. Лепихов, Л.Б. Соколинский* // Научный сервис в сети Интернет: технологии параллельного программирования. Труды Всероссийской научной конференции (18–23 сентября 2006 г., г. Новороссийск). –М.: Изд-во МГУ, 2006. -С. 39-42.

5. *Lepikhov A.V.* Data Placement Strategy in Hierarchical Symmetrical Multiprocessor Systems / *A.V. Lepikhov, L.B. Sokolinsky* // Proceedings of Spring Young Researchers' Colloquium in Databases and Information Systems (SYRCoDIS'2006), June 1-2, 2006. -Moscow, Russia: Moscow State University. -2006. –С. 31-36.
6. *А.В. Лепихов* Свидетельство Роспатента об официальной регистрации программы для ЭВМ «Параллельная СУБД «Омега» для кластерных систем» / *А.В. Лепихов, Л.Б. Соколинский, М.Л. Цымблер*; -№2008614996 от 03.10.2008.

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

Подписано в печать 16.11.2008
Формат 60x84 1/16. Бумага офсетная.
Печать офсетная. Усл. печ. л. 1,0. Уч.-изд. л. 1,2.
Тираж 100 экз. Заказ № 76. Бесплатно

Южно-Уральский государственный университет
454080 Челябинск, пр. Ленина, 76

Типография Издательства ЮУрГУ
454080, г. Челябинск, пр. им. В.И. Ленина, 76