УДК 004.657

Ю. А. Григорьев, В. Л. Плужников

МОДЕЛЬ ОБРАБОТКИ ЗАПРОСОВ В ПАРАЛЛЕЛЬНОЙ СИСТЕМЕ БАЗ ДАННЫХ

Проанализированы существующие методы выполнения запросов в параллельной СУБД. Разработана модель обработки запросов в виде замкнутой и разомкнутой систем массового обслуживания. Приведено преобразование Лапласа–Стилтьеса времени выполнения запроса в параллельной СУБД и рассмотрены варианты этого преобразования для различных архитектур параллельных систем баз данных. Проанализирован способ выбора архитектуры по критерию стоимости с ограничением на верхнюю границу доверительного интервала времени выполнения запроса.

E-mail: grigorev@iu5.bmstu.ru; plz@croc.ru

Ключевые слова: *параллельные базы данных, преобразование Лапласа–Стилтьеса, математическое ожидание времени выполнения запроса.*

Параллельные системы баз данных начинают вытеснять традиционные компьютеры основного класса, так как они позволяют работать со значительно более крупными базами данных в режиме, поддерживающем транзакции [1]. Успех таких систем опровергает прогноз статьи, опубликованной в 1983 г. [2] и предрекавшей скорое исчезновение машин баз данных. За последние десять лет компании Teradata, Tandem и другие успешно разрабатывали и продавали параллельные машины, что можно объяснить широким распространением реляционных баз данных. В 1983 г. они только еще появлялись на рынке, сегодня же доминируют. Реляционные запросы как нельзя лучше подходят для параллельного выполнения. Они состоят из однородных операций над однородным потоком данных. Каждая операция образует новое отношение (таблицу), так что из операций могут быть составлены высокопараллельные графы потоков данных.

В настоящее время не существует научно обоснованного метода оценки и выбора архитектуры параллельной системы баз данных. Сравнительный анализ архитектурных решений выполняется или на основе экспертных оценок качественных критериев (масштабируемости, доступности данных, баланса загрузки, межпроцессорных коммуникаций, когерентности кэшей, организации блокировок и др.) [3], или на основе результатов тестов (TPC и др.) для конкретных платформ [4]. Оценки времени (сложности) выполнения алгоритмов в многопроцессорных системах, как правило, представляются в виде асимп-

тотических выражений на уровне множества функций O, Ω, Θ, что не позволяет их использовать для получения численных результатов [5].

В настоящей статье предложены математические методы оценки индексов производительности и выбора архитектуры параллельных систем баз данных, учитывающие особенности выполнения запросов к базе данных проектируемой системы, а также топологию (структуру) различных архитектурных решений.

Выполнение запроса в параллельной СУБД. В этом разделе в качестве иллюстрации основных положений параллельных систем баз данных приведены некоторые сведения, изложенные Л.Б. Соколинским и М.Л. Цымблером в работе [3].

Основной формой параллельной обработки запросов является фрагментный параллелизм (рис. 1). Каждое отношение (таблица) делится на части, называемые фрагментами. Фрагменты отношения распределяются по различным процессорным узлам многопроцессорной системы. Способ фрагментации определяется функцией фрагментации ψ, которая для каждого кортежа отношения вычисляет номер процессорного узла, где должен быть размещен этот кортеж. На рис. 1 показана фрагментация отношения Поставщик (П) по атрибуту Код_П (код поставщика) на основе метода диапазонов. В данном случае функция фрагментации имеет вид: $\psi(\text{П}) = \text{П.Код\_П div 10}$; здесь div — операция деления нацело. В простейшем случае запрос параллельно выполняется на всех процессорных узлах. Полученные фрагменты сливаются в результирующее отношение.

На практике, однако, не удается избежать пересылки кортежей между процессорами во время выполнения запроса. Рассмотрим пример. Пусть отношение Поставщик (П) фрагментировано по атрибуту

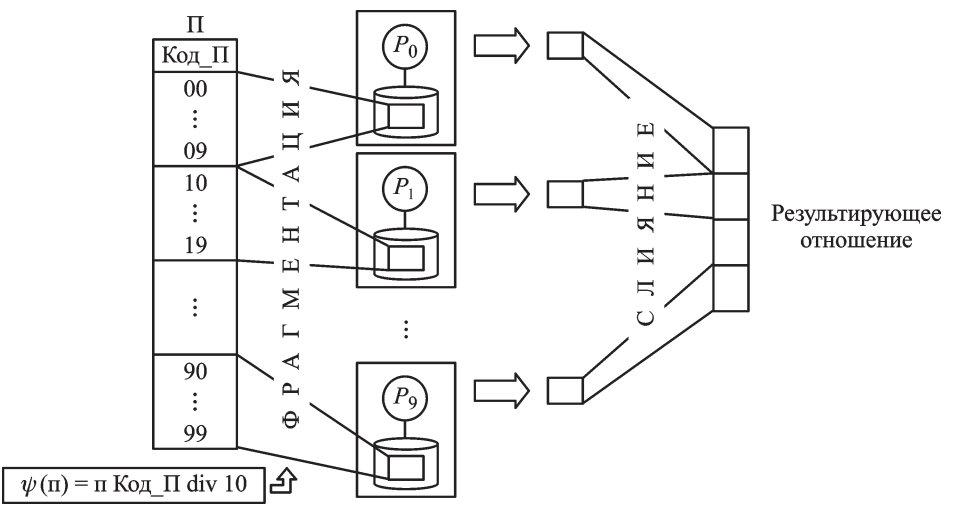

Рис. 1. Фрагментный параллелизм

Код_П, а отношение Поставщик деталей (ПД) — по атрибуту Код_Д (код детали). Предположим, что выполняется соединение отношений П и ПД в соответствии с условием П.Код_П = ПД.Код_П. При выполнении операции соединения СУБД должна перераспределять кортежи отношения ПД, так как оно фрагментировано не по атрибуту соединения. Способ перераспределения определяется функцией распределения δ, которая для каждого кортежа отношения вычисляет номер процессорного узла, на котором должен быть обработан этот кортеж. В данном примере в качестве функции распределения для отношения ПД следует взять функцию фрагментации отношения П: $\delta_{\text{ПД}}(\text{ПД}) = \psi_{\text{П}}(\text{ПД}) = \text{ПД.Код\_П div } 10$.

Если оба отношения, участвующие в соединении, фрагментированы не по атрибуту соединения, то СУБД придется перераспределять между процессорами оба входных отношения. При этом в качестве функции распределения для обоих отношений СУБД может взять любую, но одну и ту же функцию распределения по атрибуту соединения, которая отправляет кортежи с одинаковыми значениями атрибута соединения на один и тот же процессорный узел.

Для организации межпроцессорных обменов в соответствующие места дерева плана запроса СУБД вставляет специальный оператор exchange. Оператор exchange реализуется на основе использования стандартного скобочного шаблона и имеет два специальных параметра, определяемых пользователем: номер порта обмена и указатель на функцию распределения. Функция распределения для каждого кортежа вычисляет логический номер процессорного узла, на котором данный кортеж должен быть обработан. Параметр “порт обмена” позволяет включать в дерево запроса произвольное число операторов exchange (для каждого оператора указывается свой уникальный порт обмена).

Структура оператора обмена exchange изображена на рис. 2. Оператор exchange является составным и включает в себя четыре оператора: gather, scatter, split и merge, которые реализуются на базе стандартного скобочного шаблона. Оператор split — это бинарный оператор, который

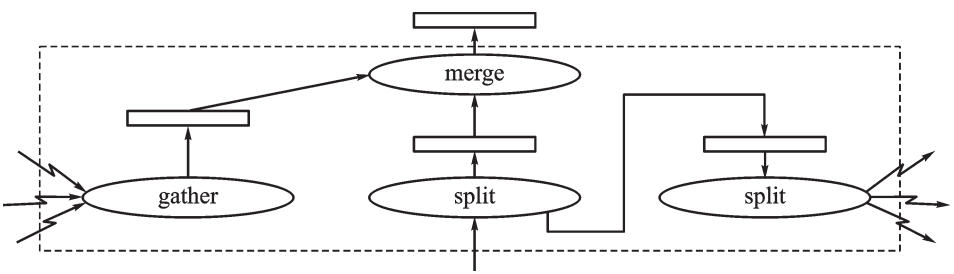

Рис. 2. Структура оператора exchange

разбивает кортежи, поступающие из входного потока, на две группы: свои и чужие. Свои кортежи — это те, которые должны быть обработаны на данном процессорном узле. Они направляются в выходной буфер оператора split. Чужие кортежи, т.е. кортежи, обработанные на процессорных узлах, отличных от данного, помещаются оператором split в выходной буфер правого дочернего узла, в качестве которого фигурирует оператор scatter. Здесь выходной буфер оператора split играет роль входного потока данных.

Нульарный оператор scatter извлекает кортежи из своего выходного буфера и пересылает их на соответствующие процессорные узлы, используя заданный номер порта. Запись кортежа в порт может быть завершена только после того, как реципиент выполнит операцию чтения из данного порта.

Нульарный оператор gather выполняет перманентное чтение кортежей из указанного порта со всех процессорных узлов, отличных от данного. Считанные кортежи помещаются в выходной буфер оператора gather.

Оператор merge определяется как бинарный оператор, который забирает кортежи из выходных буферов своих дочерних узлов и помещает их в собственный выходной буфер.

Общая схема обработки запроса в параллельной СУБД изображена на рис. 3. В соответствии с этой схемой запрос на языке SQL преобразуется в некоторый последовательный план, который, в свою очередь, преобразуется в параллельный план, представляющий собой совокупность n идентичных параллельных агентов, реализующих те же операции, что и последовательный план. Здесь n обозначает число процессорных узлов. Это достигается путем вставки оператора обмена exchange в соответствующие места дерева плана запроса. На завершающем этапе агенты рассылаются на соответствующие процессорные узлы, где интерпретируются исполнителем запросов. Результаты выполнения агентов объединяются корневым оператором exchange на нулевом процессорном модуле.

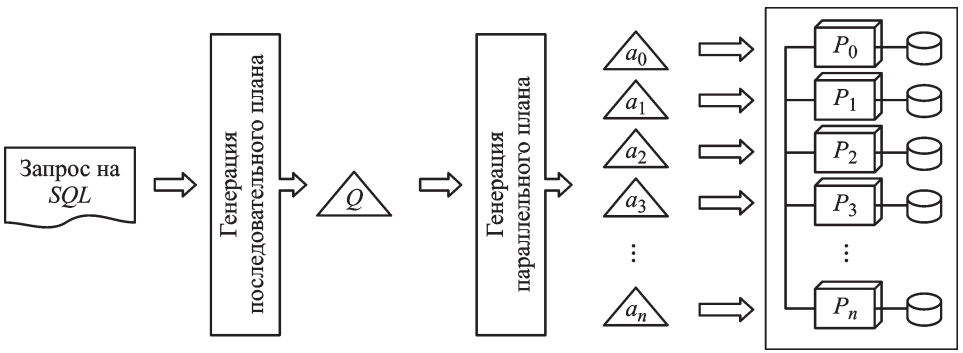

Рис. 3. Обработка запроса в параллельной СУБД

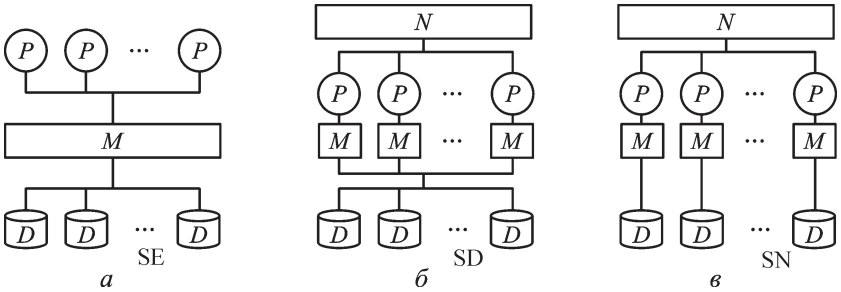

Рис. 4. Классификация Стоунбрейкера

Модель обработки запросов. Наиболее распространенной системой классификации параллельных систем баз данных является система, предложенная Майклом Стоунбрейкером [6, 7] (рис. 4). На рис. 4 введены следующие обозначения: P — процессор, M — модуль оперативной памяти (ОП), D — дисковое устройство, N — соединительная сеть. Эта классификация неполная, имеются и другие архитектуры, в частности архитектура SN может быть реализована на основе персональных компьютеров.

Далее предлагаются оценки времени выполнения запроса "select A from R where F" с планом $\pi_A(\sigma_F(R))$ для различных конфигураций параллельных систем баз данных.

На рис. 5 представлены модели обработки запроса и приняты следующие обозначения: дисциплины обслуживания: PS — Processor Sharing (квантование); IS — Immediately Served (ресурс без очереди); L — число записей в блоке БД; P_F — вероятность, что запись удовлетворяет условию поиска F; *1* — чтение блока БД с L записями с диска RAID-массива в кэш диска; *2* — перезапись блока с L записями БД из кэша диска в ОП (интенсивность λL), чтение L записей из ОП в кэш процессора (λL), сохранение записей, удовлетворяющих условию поиска F, в ОП для слияния их на выделенном процессоре ($\lambda P_F L$); *3* — обработка в процессоре L записей БД; *4* — передача записей (для SD и SN), удовлетворяющих условию поиска (с вероятностью P_F), выделенному процессору для слияния результатов (межпроцессорный обмен).

Следует отметить, что обработка данных в каком-либо ресурсе разбита на кванты. Для дискового массива — это чередующиеся секторы блока БД, для шины ОП — это слова записи БД. Для сети N — это пакеты записи БД, передаваемые по межпроцессорной шине.

Как видно из рис. 5, модель обработки запроса к БД представляет собой замкнутую СМО с различными дисциплинами обслуживания в узлах. В модели циркулируют n заявок, которые соответствуют процессорам системы. Точный метод расчета индексов производительности с помощью этой модели имеет ряд недостатков: расчеты по этой

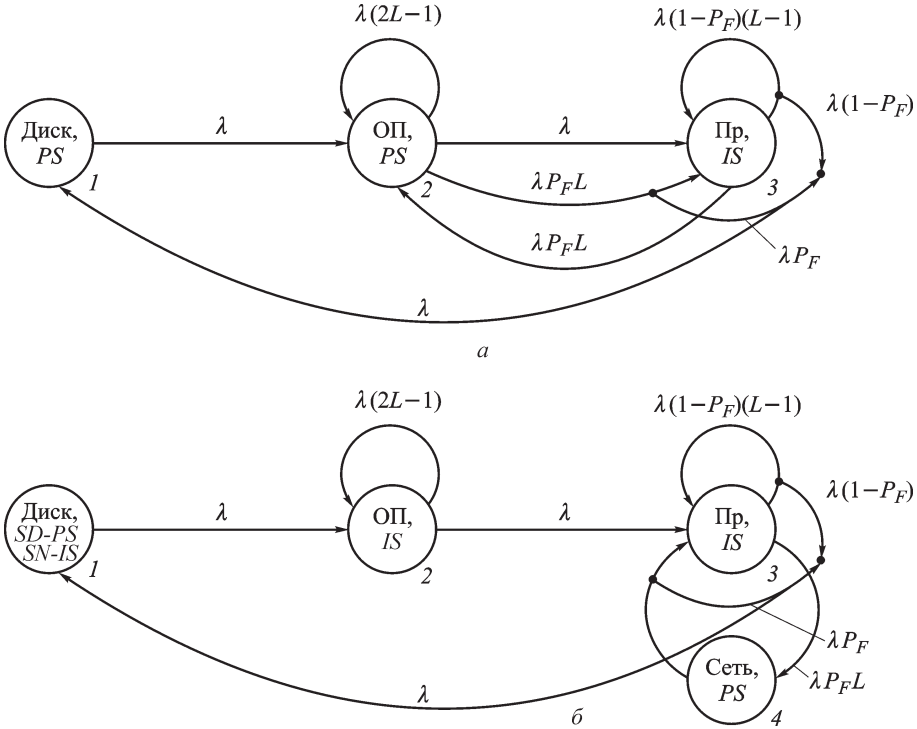

Рис. 5. Модель обработки запроса в параллельной системе баз данных с архитектурой SE (*а*) и архитектурами SD и SN (*б*)

модели достаточно сложны для большого числа процессоров n; результаты анализа нельзя представить в виде простых аналитических формул, с помощью которых можно было бы построить графики зависимостей и сравнить варианты решений.

Чтобы упростить расчеты и сделать их более наглядными, далее предложено использовать метод "узкого места". Пусть i-й и j-й узлы — это ресурсы с очередью в замкнутой СМО. Тогда из работы [8, формула (1.34)] имеем

$$\frac{\lambda_i}{\mu_i} \gg \frac{\lambda_j}{\mu_j} \Rightarrow Q_i \gg Q_j, \qquad (1)$$

где λ_i, λ_j — интенсивности входных потоков узлов; μ_i, μ_j — интенсивности обслуживания заявок в этих узлах; Q_i, Q_j — среднее число заявок в узлах.

В случае выполнения неравенства (1) для всех $j \neq i$ (т.е. при наличии "узкого места") модели (см. рис. 5) можно свести к двухузловой замкнутой СМО (рис. 6) с композиционным центром (*1*) и i-м разделяемым ресурсом (*2*). Здесь a и b — время обработки в узлах, n — число циркулирующих заявок (процессоров).

Рассматривая модели, приведенные на рис. 5, как дискретные марковские цепи в моменты выхода заявки из узлов, можно рассчитать параметры модели, представленной на рис. 6 (табл. 1).

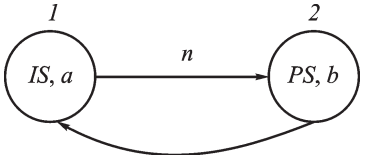

Рис. 6. Двухузловая замкнутой СМО с композиционным центром

Таблица 1

Параметры двухузловой замкнутой СМО с композиционным центром

Архитектура	Условие	Узкое место	Параметры модели	
			a	b
SE	$\frac{1}{\mu_D} \gg \frac{2+P_F}{\mu_M}$	Диск	$\left(\frac{2+P_F}{\mu_M}+\frac{1}{\mu_P}\right)L$	$\frac{1}{\mu_{DB}}$
	$\frac{2+P_F}{\mu_M} \gg \frac{1}{\mu_D}$	ОП	$\frac{1}{2+P_F}\left(\frac{1}{\mu_P}+\frac{1}{\mu_D}\right)$	$\frac{1}{\mu_M}$
SD	$\frac{1}{\mu_D} \gg \frac{P_F}{\mu_N}$	Диск	$\left(\frac{2}{\mu_M}+\frac{1}{\mu_P}+\frac{P_F}{\mu_N}\right)L$	$\frac{1}{\mu_{DB}}$
	$\frac{P_F}{\mu_N} \gg \frac{1}{\mu_D}$	Сеть	$\frac{1}{P_F}\left(\frac{1}{\mu_P}+\frac{1}{\mu_D}+\frac{2}{\mu_M}\right)$	$\frac{1}{\mu_N}$
SN	Нет	Сеть	$\frac{1}{P_F}\left(\frac{1}{\mu_P}+\frac{1}{\mu_D}+\frac{2}{\mu_M}\right)$	$\frac{1}{\mu_N}$

В табл. 1 приняты следующие обозначения: μ_D — интенсивность чтения записей БД с диска RAID-массива; $\mu_D = \mu_{DB} \cdot L$, где μ_{DB} — интенсивность чтения блоков БД с диска; μ_M — интенсивность чтения/сохранения записей БД в ОП; μ_N — интенсивность передачи записей БД по сети (межпроцессорный обмен); μ_P — интенсивность обработки записей БД в процессоре.

Сведение замкнутой двухузловой СМО к разомкнутой. Модель, приведенная на рис. 6, проще, чем модели, представленные на рис. 5. Однако она имеет существенный недостаток: результаты анализа нельзя представить в виде простых аналитических формул, с помощью которых можно было бы сравнить варианты решений.

Рассмотрим два случая (см. рис. 6).

1. *Загрузка ресурса 2 большая*. В этом случае интенсивность выходного потока примерно равна $1/b$. Тогда средняя длина очереди в разделяемом ресурсе *2* составляет $k = n - a/b$. Отсюда получаем

$$\frac{nb}{a} = \frac{n}{n-k} > 1. \qquad (2)$$

В этом случае можно сделать интересный вывод. Пусть $b > a$ (разделяемый ресурс *2* медленный). Тогда $Q_2 > Q_1$, где Q_2 и Q_1 —

среднее число заявок в ресурсах *2* и *1*. Это следует из утверждения [8, формула (1.34)]

$$b^i > \frac{a^i}{i!}, \quad i = 1, \ldots, n \Rightarrow Q_2 > Q_1. \tag{3}$$

Значит, $Q_2 = \tau n$, $\frac{1}{2} < \tau < 1$. Отсюда получаем оценку для среднего времени обработки записей таблицы базы данных

$$T = (bQ_2 + a)\frac{S}{n} = Sb\Big(\tau + \frac{a}{nb}\Big), \tag{4}$$

где S — либо число блоков в таблице ("узкое место" — диск), либо число записей в таблице ("узкое место" — ОП или сеть).

Из уравнения (4) следует, что T слабо зависит от числа процессоров n. Отсюда можно сделать следующий вывод: если в системе имеется медленный разделяемый ресурс, то распараллеливание выполнения запроса по нескольким процессорам не приведет к существенному уменьшению времени обработки этого запроса к БД.

2. *Загрузка ресурса 2 небольшая*. Интенсивность выходного потока равна P/b, где P — вероятность, что разделяемый ресурс *2* занят. Отсюда для достаточно больших n имеем

$$\rho = \frac{nb}{a} = \frac{Pn}{n-k} < 1. \tag{5}$$

Известно [8], что СМО (см. рис. 6) с дисциплиной квантования (PS) эквивалентна СМО с дисциплиной "первый пришел — первый обслужен" (FCFS) и экспоненциальным распределением времени обслуживания (по крайней мере, это справедливо для распределения числа заявок в разделяемом ресурсе *2* и среднего времени пребывания в нем). Для последней СМО справедливо следующее распределение вероятностей числа заявок в разделяемом ресурсе *2* [9]:

$$P_i = P_0\Big(\frac{nb}{a}\Big)^i \frac{(n-i+1)(n-i+2)\ldots n}{n^i}, \quad 1 \leqslant i \leqslant n. \tag{6}$$

При малых i и больших n

$$P_i \to P_0\Big(\frac{nb}{a}\Big)^i. \tag{7}$$

Если выполняется условие (5), то $P_0 \to 1 - \rho = 1 - \frac{nb}{a}$. Ошибка вычисления P_i по формуле (7) возрастает при увеличении i, но в этом случае P_i мало.

Однако распределение (7) справедливо для разомкнутой СМО M/M/1. Интенсивность входного потока для M/M/1 определяется по

формуле

$$\Lambda = n\lambda, \quad \lambda = \frac{1}{a}, \tag{8}$$

где параметр a рассчитывается по формулам, приведенным в табл. 1.

Далее будем использовать следующие обозначения для интенсивностей входного потока:

λ_{DB} и $\lambda_D = \lambda_{DB}L$ — интенсивность заявок на чтение с диска от одного процессора блоков и соответственно записей БД; λ_M — интенсивность заявок на чтение/сохранение записей БД в ОП от одного процессора; λ_N — интенсивность заявок на передачу записей БД по сети межпроцессорного обмена от одного процессора.

Можно сформулировать стратегию вычислений индексов производительности параллельной системы базы данных, построенной на основе какого-либо архитектурного решения (см. рис. 4):

1. Выявить "узкое место" системы (см. табл. 1).

2. Если выполняется неравенство (2), то считать архитектуру параллельной системы базы данных неудачной и перейти к другому архитектурному решению.

3. Если выполняется неравенство (5), то для расчетов модели, приведенной на рис. 6, использовать разомкнутую СМО M/M/1 с параметрами, которые указаны в табл. 1.

Оценка времени выполнения запроса. Используя подход, предложенный в работе [10], можно вывести следующее преобразование Лапласа–Стилтьеса (ПЛС) времени выполнения запроса к базе данных с планом $\pi_A(\sigma_F(R))$:

$$\phi(s) = G(\phi_D^{1/L}(s)\phi_M^2(s)(1 - P_F(1 - \phi_N(s)))\phi_P(s)), \tag{9}$$

где $G = z^{V/n}$ — производящая функция числа записей фрагментированной таблицы R, обрабатываемых на одном процессоре; V — общее число записей в таблице R; n — число процессоров (или персональных компьютеров в кластере); $\phi_D(s)$ — ПЛС времени чтения блока БД фрагментированной таблицы с диска (с учетом очереди к дисковому массиву), L — число записей таблицы в блоке БД; $\phi_M(s)$ — ПЛС времени чтения/сохранения записи фрагментированной таблицы в оперативной памяти (с учетом очереди к шине памяти); $\phi_N(s)$ — ПЛС времени межпроцессорного обмена при передаче результирующей записи по сети N; $\phi_P(s)$ — ПЛС времени обработки записи в процессоре, который является неразделяемым ресурсом (будем предполагать, что это время распределено по экспоненциальному закону); P_F — вероятность, что запись удовлетворяет условию поиска F (эта вероятность рассчитывается по известным формулам [10]) .

На основе приведенных рассуждений будем считать, что каждое ПЛС $\phi_D(s)$, $\phi_M(s)$, $\phi_N(s)$ соответствует времени пребывания в СМО

M/M/1. Время пребывания в СМО M/M/1 распределено по экспоненциальному закону. Для $\phi_D(s)$, $\phi_M(s)$, $\phi_N(s)$, $\phi_P(s)$ имеем

$$\phi_D(s) = \frac{(\mu_D - n\lambda_D)(\mu_D + Ls)}{\mu_D(\mu_D - n\lambda_D + Ls)} \cdot \frac{\mu_D}{(\mu_D + Ls)} = \phi_{DW}(s)\phi_{DC}(s); \quad (10)$$

$$\phi_M(s) = \frac{(\mu_M - n\lambda_M)(\mu_M + s)}{\mu_M(\mu_M - n\lambda_M + s)} \cdot \frac{\mu_M}{(\mu_M + s)} = \phi_{MW}(s)\phi_{MC}(s); \quad (11)$$

$$\phi_N(s) = \frac{(\mu_N - n\lambda_N)(\mu_N + s)}{\mu_N(\mu_N - n\lambda_N + s)} \cdot \frac{\mu_N}{(\mu_N + s)} = \phi_{NW}(s)\phi_{NC}(s); \quad (12)$$

$$\phi_P(s) = \frac{\mu_P}{\mu_P + s}. \quad (13)$$

Здесь n — число процессоров; $\phi_{.C}(s)$, $\phi_{.W}(s)$ — это соответственно ПЛС времени обработки записи и ожидания обработки ее в ресурсе.

Обозначения μ и λ были введены ранее; при этом должно выполняться неравенство

$$\rho = \frac{n\lambda}{\mu} < 1. \quad (14)$$

Преобразования Лапласа–Стилтьеса времени выполнения запроса к базе данных (см. (9)) для различных конфигураций (см. рис. 4) отличаются присутствием или отсутствием в них ПЛС $\phi_{DW}(s)$, $\phi_{MW}(s)$, $\phi_{NW}(s)$ (табл. 2). Все зависит от того, разделяемый ли в конфигурации ресурс (т.е. возможна ли к нему очередь) или нет. Если ПЛС отсутствует, то в формулах (10)–(12) его заменяют единицей.

Наличие составляющей $\phi_{.W}(s)$ зависит также от того, является ли ресурс “узким местом” (см. табл. 1).

Из выражения (9) можно получить математическое ожидание и дисперсию времени выполнения запроса к БД:

$$M_\xi = -\phi'(0); \quad (15)$$

$$M_{\xi^2} = \phi^{(2)}(0); \quad (16)$$

$$\sigma_\xi^2 = M_{\xi^2} - M_\xi^2. \quad (17)$$

В качестве примера выведем теперь выражение для математического ожидания времени выполнения запроса к БД для архитектуры SE в предположении, что “узким местом” является диск (см. табл. 1).

Таблица 2

Наличие/отсутствие ПЛС в формуле (9)

Архитектура	$\phi_{DW}(s)$	$\phi_{MW}(s)$	$\phi_{NW}(s)$
SE	Да	Да	$\phi_N(s)$ следует заменить на $\phi_M(s)$, так как нет сети
SD	Да	Нет	Да
SN	Нет	Нет	Да

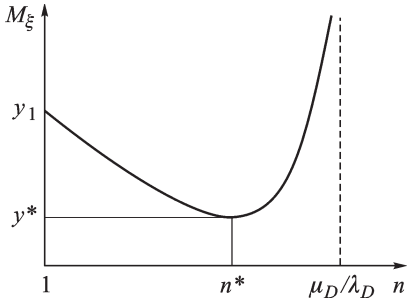

Рис. 7. График зависимости M_ξ от числа процессоров n

Дифференцируя выражение (9) как сложную функцию в нуле, получаем

$$M_\xi = -\phi'(0) = \frac{V}{n}\Big(\frac{1}{\mu_D - n\lambda_D} + \frac{1}{\mu_{MP}}\Big),$$

$$\frac{1}{\mu_{MP}} = \frac{\mu_M + \mu_P(2 + P_F)}{\mu_M \mu_P}. \qquad (18)$$

С помощью формул (16) и (17) можно записать выражения для M_{ξ^2} и σ^2_ξ.

График зависимости математического ожидания M_ξ времени выполнения запроса от числа процессоров n в параллельной системе баз данных показан на рис. 7.

Координаты точек, обозначенных на рис. 7 (n^* — точка минимума), вычисляются по формулам:

$$y_1 = M_\xi(1) = V(\frac{1}{\mu_D - \lambda_D} + \frac{1}{\mu_{MP}}); \qquad (19)$$

$$n^* = \frac{\mu_D}{\lambda_D} - \frac{\mu_{MP}}{\lambda_D}\Big(\sqrt{\frac{\mu_D}{\mu_{MP}} + 1} - 1\Big); \qquad (20)$$

$$y^* = M_\xi(n^*) = \frac{V}{n^*}\Big(\frac{1}{\mu_D - n^*\lambda_D} + \frac{1}{\mu_{MP}}\Big). \qquad (21)$$

Из уравнения (20) следует, что при возрастании μ_{MP} значение n^* убывает. Используя правило Лопиталя, запишем

$$\lim_{\mu_{MP}\to\infty} n^* = \frac{\mu_D}{2\lambda_D}. \qquad (22)$$

Можно показать, что производная n^* по P_F больше нуля. Поэтому значение n^* возрастает с увеличением P_F ($0 \leqslant P_F \leqslant 1$).

Поведение функции $M_\xi(n)$ (см. рис. 7) объясняется тем, что сначала с ростом числа процессоров время убывает благодаря распараллеливанию обработки запроса, затем время возрастает из-за перегрузки подсистемы ввода/вывода.

Метод выбора архитектуры системы. Используя сведения, приведенные в табл. 1 и 2, а также формулы (9)–(17), можно получить математические ожидания и дисперсии времени выполнения запроса к БД для всех архитектур, представленных на рис. 4. С помощью этих выражений можно выполнить сравнение архитектур по стоимости с учетом ограничения на время выполнения запроса или смеси запросов.

Для каждой i-й архитектуры необходимо решить задачу оптимизации:

$$C_i(n) \underset{n}{\to} \min$$

$$M_{\xi i} + k\sigma_{\xi i} < T_{\text{гр}}, \tag{23}$$

где $C_i(n)$ — стоимость i-й архитектуры с n процессорами; $M_{\xi i} + k\sigma_{\xi i}$ — верхняя граница доверительного интервала ($k = 1 \dots 3$); $T_{\text{гр}}$ — предельно допустимое время выполнения запроса или смеси запросов.

Следует отметить, что если величина V/n достаточна велика, то в соответствии с центральной предельной теоремой Ляпунова функция распределения времени выполнения запроса стремится к нормальному закону (это объясняет граничное значение k, равное 3).

Оптимальной будет архитектура с минимальным значением $C_i(n_{opt\,i})$. Смысл задачи (23) заключается в том, что какая-то архитектура может быть быстрой, но иметь высокую стоимость, и наоборот.

Заключение. 1. Проанализирован существующий способ выполнения запросов в параллельной системе баз данных. СУБД выполняет фрагментацию таблиц (отношений) базы данных и пересылку фрагментов процессорам. Оптимальный последовательный план тиражируется по процессорам, т.е. создаются n идентичных параллельных агентов, каждый из которых выполняет операции с фрагментами таблиц, переданными процессору. Результаты, полученные от всех процессоров, объединяются.

2. Разработана модель обработки запросов в параллельной системе баз данных в виде замкнутой и разомкнутой СМО, учитывающая наличие “узкого места” в системе.

3. Приведено преобразование Лапласа–Стилтьеса времени выполнения запроса, имеющего план $\pi_A(\sigma_F(R))$, в параллельной СУБД. Рассмотрены варианты этого преобразования для различных архитектур параллельных систем баз данных, т.е. ПЛС времени ожидания освобождения ресурса входит или не входит в исходное ПЛС времени выполнения запроса в зависимости от того, является ли ресурс разделяемым или нет.

4. Исследована зависимость математического ожидания времени выполнения запроса к БД от числа процессоров для архитектуры SE. Показано, что сначала с ростом числа процессоров время убывает благодаря распараллеливанию обработки запроса, затем время возрастает из-за перегрузки подсистемы ввода/вывода, которая является разделяемым ресурсом.

5. Предложен способ выбора архитектуры параллельной системы баз данных по критерию стоимости с ограничением на верхнюю границу доверительного интервала времени выполнения запроса или смеси запросов.

6. Предполагается продолжить исследования и получить оценки времени выполнения запросов с более сложными планами реализации (например, для плана соединения таблиц).

СПИСОК ЛИТЕРАТУРЫ

1. Дэвид Девитт, Джим Грэй. Параллельные системы баз данных: будущее высокоэффективных систем баз данных / Пер. С. Кузнецова, 2009: [Электронный ресурс]. [http://www.citforum.ru/database/classics/parallel_f]. Проверено 10.04.2009.
2. Boral H. and DeWitt D. Database machines: An idea whose time has passed? A critique of the future of the database machines. In Proceedings of the 1983 Workshop on Database Machines. H.-O. Leilich and M. Missikoff, Eds., Springer-Verlag, 1983.
3. Соколинский Л. Б., Цымблер М. Л. Лекции по курсу "Параллельные системы баз данных": [Электронный ресурс]. [http://pdbs.susu.ru/CourseManual.html]. Проверено 10.04.2009.
4. Transaction Processing Performance Council (TPC): [Электронный ресурс]. [http://www.tpc.org]. Проверено 10.04.2009.
5. Миллер Р., Боксер Л. Последовательные и параллельные алгоритмы. Общий подход. – М.: БИНОМ. Лаборатория знаний, 2006. – 406 с.
6. Соколинский Л. Б. Обзор архитектур параллельных систем баз данных // Программирование. – 2004. – № 6. – С. 1–15.
7. David Dewitt, Jim Gray. Parallel database systems: the future of high performance database systems // Communications of the ACM. – June 1992. – Vol. 35. No. 6. – P. 1–26.
8. Жожикашвили В. А., Вишневский В. М. Сети массового обслуживания. Теория и применение к сетям ЭВМ. – М.: Радио и связь, 1988. – 192 с.
9. Клейнрок Л. Теория массового обслуживания. – М.: Машиностроение, 1979. – 432 с.
10. Григорьев Ю. А., Плутенко А. Д. Теоретические основы анализа процессов доступа к распределенным базам данных. – Новосибирск: Наука, 2002. – 180 с.

Статья поступила в редакцию 22.03.2010

Юрий Александрович Григорьев родился в 1951 г., окончил МГТУ им. Н.Э. Баумана в 1975 г. Д-р техн. наук, профессор кафедры "Системы обработки информации и управления" МГТУ им. Н.Э. Баумана. Автор более 75 научных работ в области обработки информации и управления.

Yu.A. Grigoriev (b. 1951) graduated from the Bauman Moscow Higher Technical School in 1975. D. Sc. (Eng.), professor of "Systems of Data Processing and Management" department of the Bauman Moscow State Technical University. Author of more than 75 publications in the field of data processing and management.

Всеволод Львович Плужников родился в 1984 г., окончил МГТУ им. Н.Э. Баумана в 2006 г. Ведущий инженер ЗАО "КРОК Инкорпорейтед". Автор одной научной работы в области обработки информации и управления.

V.L. Pluzhnikov (b. 1984) graduated from the Bauman Moscow State Technical University in 2006. Leading engineer of private company "KROK Incorporated". Author of 1 publication in the field of data processing and management.