электронное научно-техническое издание
НАУКА и ОБРАЗОВАНИЕ
Эл № ФС 77 - 30569. Государственная регистрация №0421100025. ISSN 1994-0408

Оценка времени выполнения запроса в параллельной системе баз данных

 # 06, июнь 2011
автор: Плужников В. Л.
УДК 004.657

МТГУ им. Н. Э. Баумана
vpluzhnikov@gmail.com

Введение

На сегодняшний день параллельные системы баз данных часто применяются для обработки больших объемов данных. Параллельная обработка разделяет большие задачи для обработки на множество маленьких задач, которые можно выполнять параллельно на нескольких узлах. Результатом такого разбиения, как правило, является уменьшение времени выполнения первичной, большой задачи. 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, или на основе результатов тестов для конкретных платформ [1,3,7].

В статье предлагается математический метод оценки и выбора архитектуры параллельных систем баз данных, учитывающий особенности выполнения запросов к базе данных проектируемой системы, а также топологию (структуру) различных архитектурных решений.

Особенности выполнения запроса в параллельной СУБД

Рассмотрим процесс выполнения SQL запросов в параллельной системе управления базами данных [1].

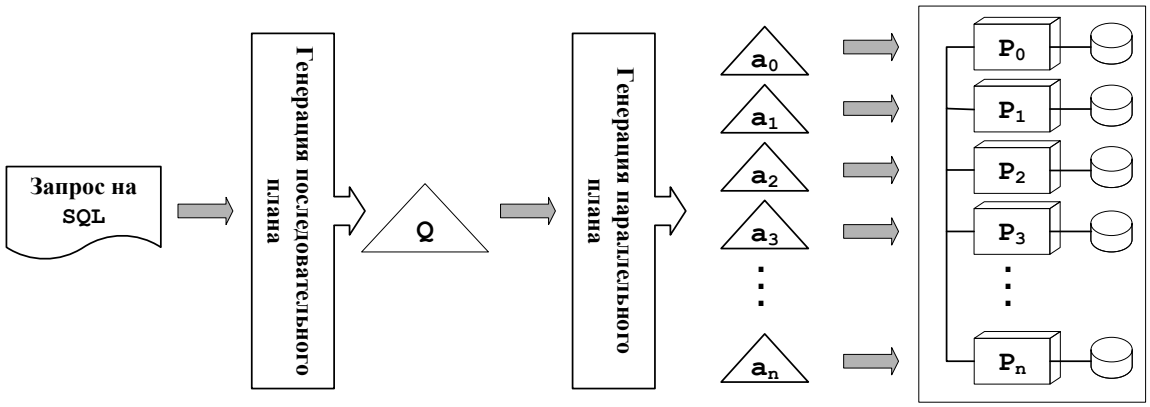


Рис. 1. Обработка запроса в параллельной СУБД.

Общая схема обработки запроса в параллельной СУБД изображена на рис. 1. SQL запрос, в соответствии со схемой выполняется в несколько этапов [2]:

Этап 1. Генерация последовательного плана выполнения SQL запроса.

Этап 2. Тиражирование плана выполнения запроса на все процессоры системы.

Этап 3. Обработка запроса над фрагментированными таблицами

Этап 4. Слияние полученных данных.

Для организации межпроцессорных обменов в соответствующие места дерева плана запроса СУБД вставляет специальный оператор exchange. Оператор exchange реализуется на основе использования стандартного скобочного шаблона. Оператор exchange 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 количество операторов exchange (для каждого оператора указывается свой уникальный порт обмена).

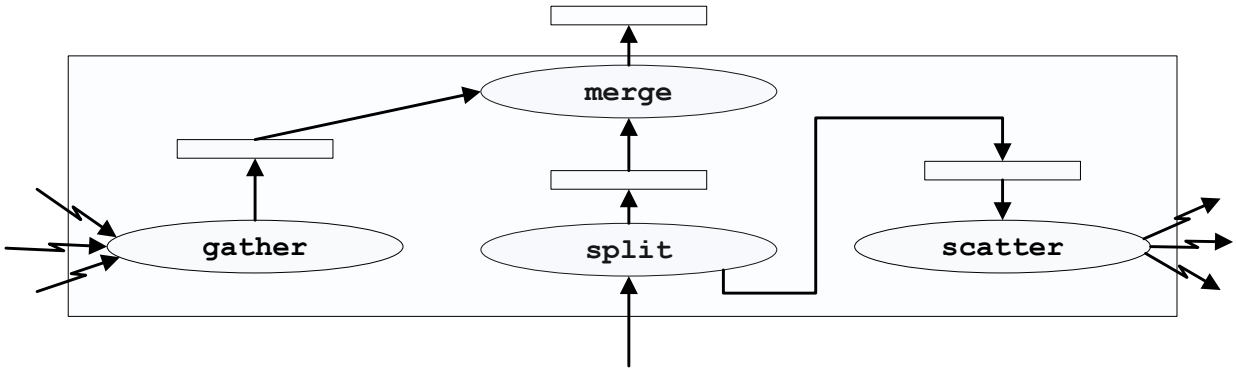


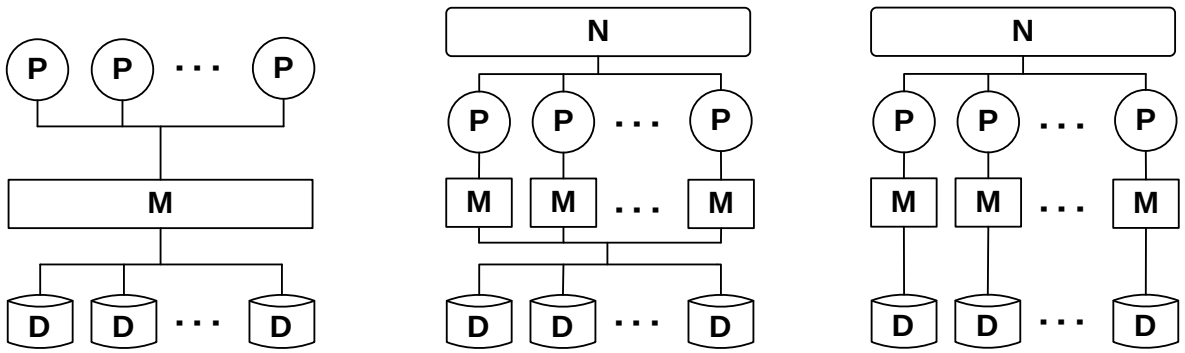
Рис. 2. Структура оператора exchange.

Структура оператора обмена exchange изображена на рис. 2.

Оператор **exchange** является составным оператором и включает в себя четыре оператора: gather, scatter, split и merge. Оператор split – это бинарный оператор, который осуществляет разбиение кортежей, поступающих из входного потока, на две группы: свои и чужие. Свои кортежи – это кортежи, которые должны быть обработаны на данном процессорном узле. Эти кортежи направляются в выходной буфер оператора split. Чужие кортежи, то есть кортежи, которые должны быть обработаны на процессорных узлах, отличных от данного, помещаются оператором split в выходной буфер правого дочернего узла, в качестве которого фигурирует оператор scatter. Нульарный оператор scatter извлекает кортежи из своего выходного буфера и пересылает их на соответствующие процессорные узлы. Нульарный оператор gather выполняет перманентное чтение кортежей из указанного порта со всех процессорных узлов, отличных от данного. Оператор merge определяется как бинарный оператор, который забирает кортежи из выходных буферов своих дочерних узлов и помещает их в собственный выходной буфер. На базе данных операций оператор exchange реализует полноценный межпроцессорный обмен записями в параллельной СУБД при обработке запроса методом фрагментарного параллелизма.

Оценка времени выполнения запроса в параллельной системе баз данных

Наиболее распространенной системой классификации параллельных систем баз данных является система, предложенная Майклом Стоунбрейкером (Michael Stonebraker) [2,3], показанная на рис. 3. Здесь P обозначает процессор, M – модуль оперативной памяти, D – дисковое устройство, N – соединительную сеть. Эта классификация неполная, имеются и другие конфигурации. В частности архитектура SN может быть реализована на основе персональных компьютеров.



a) SE **a) SD** **a) SN**

Рис. 3. Классификация Стоунбрейкера.

Используя подход, предложенный в [4], можно вывести следующее преобразование Лапласа-Стилтьеса (ПЛС) времени выполнения запроса к базе данных с планом $\pi_A(\sigma_F(R))$:

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

где

$G = z^{V/n}$ – производящая функция числа записей фрагментной 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 [4]) .

Будем считать, что каждое из преобразований Лапласа-Стилтьеса $\phi_D(s)$, $\phi_M(s)$, $\phi_N(s)$ соответствует времени пребывания в СМО M/M/1.

Действительно, обработку в ресурсе (дисковом массиве, оперативной памяти, сети) можно описать с помощью модели ремонтника [6], где в качестве приборов выступают процессоры, а в качестве ремонтника – ресурс (шина). При достаточно большом числе процессоров эту модель можно аппроксимировать разомкнутой СМО M/G/1 при загрузке ремонтника $\rho < 1$.

Обработка записи базы данных в ресурсе разбита на кванты. Для дискового массива – это поиск в кэше файловой системы, обработка запроса адаптером шины сервера, поиск в кэше контроллера дискового массива, поиск в кэше диска, подвод головки к цилиндру, ожидание требуемого сектора, чтение сектора в кэш диска, запись сектора в кэш контроллера дискового массива, запись сектора в кэш файловой системы. Для шины оперативной памяти – это чтение/запись большого количества операндов и

команд, выполняемых в процессоре при обработке записи базы данных. Для шины N – это передача большого числа сообщений протокольной машины.

Известно [5], что СМО M/G/1 с дисциплиной квантования приближается по характеристикам к СМО M/M/1 при малом значении интервала квантования (по крайней мере, это справедливо для распределения числа заявок в СМО и среднего времени пребывания).

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:

$$\phi_D(s) = \frac{(\mu_D - n\lambda_D)(\mu_D + Ls)}{\mu_D(\mu_D - n\lambda_D + Ls)} \cdot \frac{\mu_D}{(\mu_D + Ls)} = \phi_{DW}(s) \cdot \phi_{DC}(s), \quad (2)$$

$$\phi_M(s) = \frac{(\mu_M - n\lambda_M)(\mu_M + s)}{\mu_M(\mu_M - n\lambda_M + s)} \cdot \frac{\mu_M}{(\mu_M + s)} = \phi_{MW}(s) \cdot \phi_{MC}(s), \quad (3)$$

$$\phi_N(s) = \frac{(\mu_N - n\lambda_N)(\mu_N + s)}{\mu_N(\mu_N - n\lambda_N + s)} \cdot \frac{\mu_N}{(\mu_N + s)} = \phi_{NW}(s) \cdot \phi_{NC}(s), \quad (4)$$

$$\phi_P(s) = \frac{\mu_P}{\mu_P + s}. \quad (5)$$

Здесь

n - число процессоров,

$\phi_{.C}(s)$, $\phi_{.W}(s)$ – это соответственно ПЛС времени обработки записи и ожидания обработки её в ресурсе.

Обозначения для μ и λ введены раньше (см. предыдущий раздел). При этом должно выполняться неравенство

$$\rho = \frac{n\lambda}{\mu} < 1. \quad (6)$$

ПЛС времени выполнения запроса к базе данных (см. (1)) для различных конфигураций (см. рис. 3) отличаются присутствием или отсутствием в них ПЛС $\phi_{DW}(s)$, $\phi_{MW}(s), \phi_{NW}(s)$ (табл. 1). Всё зависит от того, разделяемый ли в конфигурации ресурс (т.е. возможна к нему очередь) или нет. Если преобразование Лапласа-Стилтьеса отсутствует, то в формулах (2)-(4) оно заменяется единицей (в табл. 2 это отмечено символом '-').

Таблица 1

Наличие (+) или отсутствие (-) ПЛС $\varphi_{DW(S)}$, $\varphi_{MW(S)}$, $\varphi_{NW(S)}$ в формуле (1) для различных конфигураций

	$\varphi_{DW(S)}$	$\varphi_{MW(S)}$	$\varphi_{NW(S)}$
SE	+	+	$\varphi_{N(S)}$ следует заменить на $\varphi_{M(S)}$, т.к. нет сети
SD	+	-	+
SN	-	-	+

Из (1) можно получить математическое ожидание и дисперсию времени выполнения запроса к БД:

$$M_{\xi} = -\phi'(0), \tag{7}$$

$$M_{\xi^2} = \phi^{(2)}(0), \tag{8}$$

$$\sigma_{\xi}^2 = M_{\xi^2} - M_{\xi}^2. \tag{9}$$

Получим теперь выражение для математического ожидания времени выполнения запроса к БД для конфигураций SE, SD и SN. Предположим, что "узким местом" в архитектурах SE и SD является диск (см. табл. 1).

Дифференцируя (1) как сложную функцию в нуле, получим для конфигураций SE, SD и SN соответственно:

$$M_{\xi SE} = -\phi'(0) = \frac{V}{n}\left(\frac{1}{\mu_D - n\lambda_D} + \frac{2 + P_F}{\mu_M} + \frac{1}{\mu_P}\right), \tag{10}$$

$$M_{\xi SD} = -\phi'(0) = \frac{V}{n}\left(\frac{1}{\mu_D - n\lambda_D} + \frac{2}{\mu_M} + \frac{1}{\mu_P} + \frac{P_F}{\mu_N}\right), \tag{11}$$

$$M_{\xi SN} = -\phi'(0) = \frac{V}{n}\left(\frac{1}{\mu_D} + \frac{2}{\mu_M} + \frac{1}{\mu_P} + \frac{P_F}{\mu_N - n\lambda_N}\right), \tag{12}$$

С помощью формул (8) и (9) можно получить выражения для M_{ξ^2} и σ_{ξ}^2.

Пример расчета времени выполнения запроса в параллельной СУБД

Расчёт математического ожидания (среднего) времени выполнения запроса к таблице в ПСБД был выполнен при следующих значениях характеристик ресурсов.

1. Процессор – IntelXeonE5310. СУБД – Oracle11g. При расчете примера была выполнена автотрассировка запроса: число обработанных записей в таблице – 50000, число процессорных циклов – $3.9 \cdot 10^8$. Для выбранного процессора в системе приведено значение числа процессорных циклов, выполняемых Oracle в секунду (CPUSPEEDNW) – $2.6 \cdot 10^9$. Поэтому для интенсивности обработки записей в процессоре имеем – $\mu_P = 5.3 \cdot 10^4/(3.9 \cdot 10^8/2.6 \cdot 10^9) = 3{,}5 \cdot 10^5$.

2. Внешняя память – RAID10 с $К_Д$=10 дисками 3.5" SeagateCheetah 15K.6 ST3146356FC; размер блока чередования (stripesize) – $Q_{БЧ}$=256 Кб; среднее время поиска и чтения блока чередования с диска – $t_{БЧ} = t_{подвода} + t_{вращения}/2 + Q_{БЧ}/v_{чтения} = 4 + 4/2 + 256/200 = 7$ мс.

Размер блока СУБД – $Q_{БС}$=16Кб; средняя длина записи таблиц A и B – $l_З$=0.256Кб, размер многоблочного чтения СУБД (переменная СУБД db_file_multiblock_read_count) – $q_{МБ}$=80. Поэтому для интенсивности чтения записей БД из массива RAID имеем – $\mu_D = q_{МБ} \cdot Q_{БС}/l_З/(\lceil q_{МБ} \cdot Q_{БС}/Q_{БЧ}/(К_Д/2)\rceil \cdot t_{БЧ}) = 0.7 \cdot 10^6$. Число 2 в этой формуле учитывает тот факт, что при чтении полосы RAID-массива (т.е. при одновременном чтении блоков чередования с разных дисков) используется половина дисков массива, так как в RAID10 каждый диск из этой половины имеет зеркальную копию.

3. Оперативная память – DDR2-667DPC2- 5300. Расчёты показывают, что интенсивность чтения записей базы данных из ОП равна $\mu_M = 5.2 \cdot 10^6$.

4. Высокоскоростная системная шина: для SD - ParallelSysplexInfiniband12xQDR [8], для SN – ByNet. В обоих случаях узлы соединены по схеме "точка-точка" (т.е. шина является неразделяемым ресурсом) [9] и интенсивность передачи записей по этим шинам равна $\mu_N = 50.3 \cdot 10^6$.

Далее приведены значения остальных параметров: число записей в таблице $V=10^6$, вероятность фильтрации записей таблицыP_F=0,05.

При расчётах использовались формулы (10)-(12). Для архитектур SE и SD "узким местом" является внешняя память (диск). При вычислениях величину '$n\lambda_D$' необходимо

разделить на 2, так как при обработке запросов на чтение от разных процессоров может быть использована не только основная, но и зеркальная половина дисков. Для архитектуры SN произведение '$n\lambda_N$' необходимо исключить из формулы расчёта математического ожидания (т.е. обнулить '$n\lambda_N$' в колонке 6 табл. 3 для строки SN), так как в рассматриваемом примере шина является неразделяемым ресурсом. Графики зависимости математического ожидания времени выполнения соединения двух таблиц от числа процессоров 'n' в параллельной системе баз данных приведены на рис. 4 и 5.

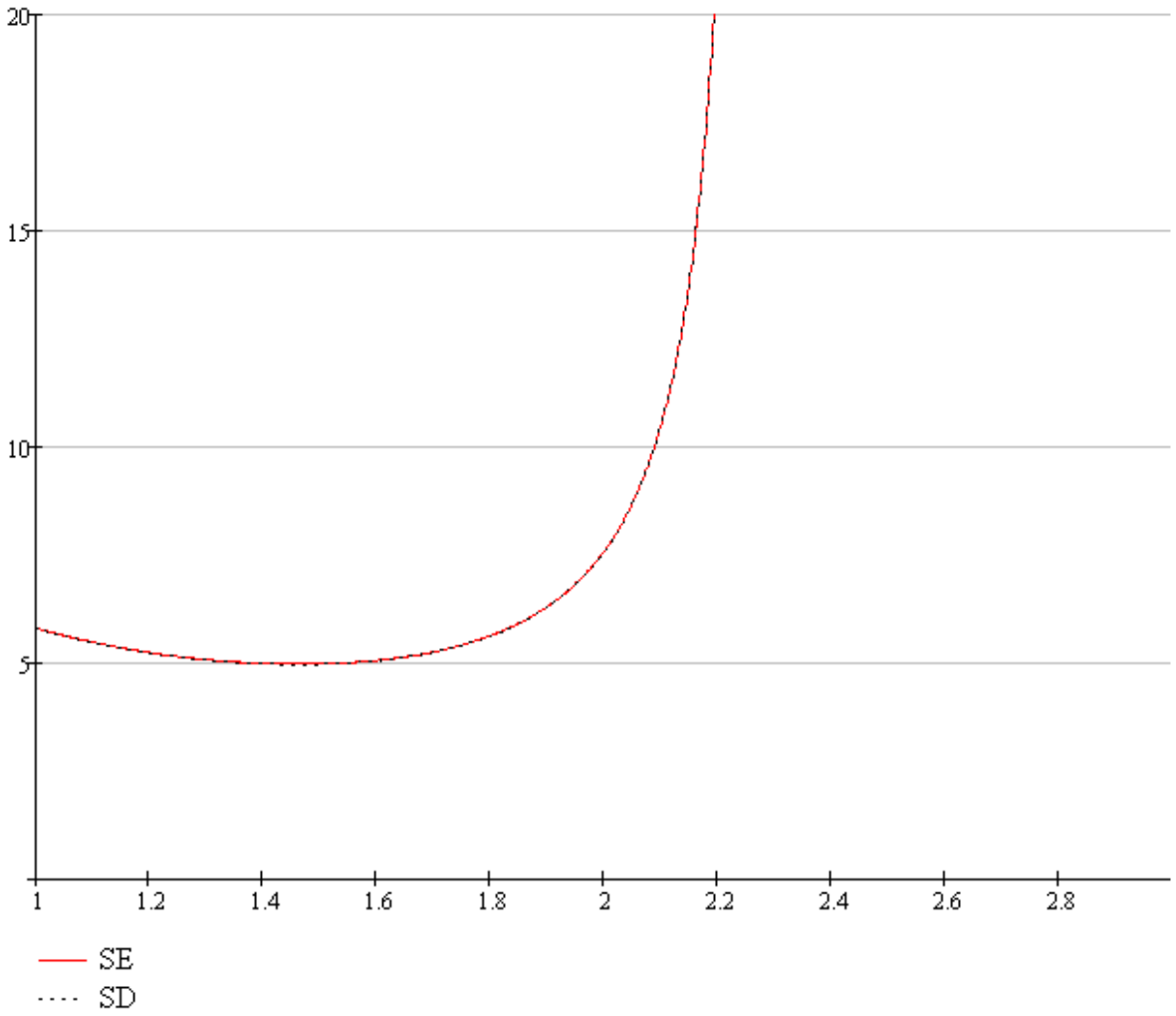


Рис. 4. Зависимость математического ожидания времени выполнения запроса к таблицам архитектурах SE, SD от числа процессоров.

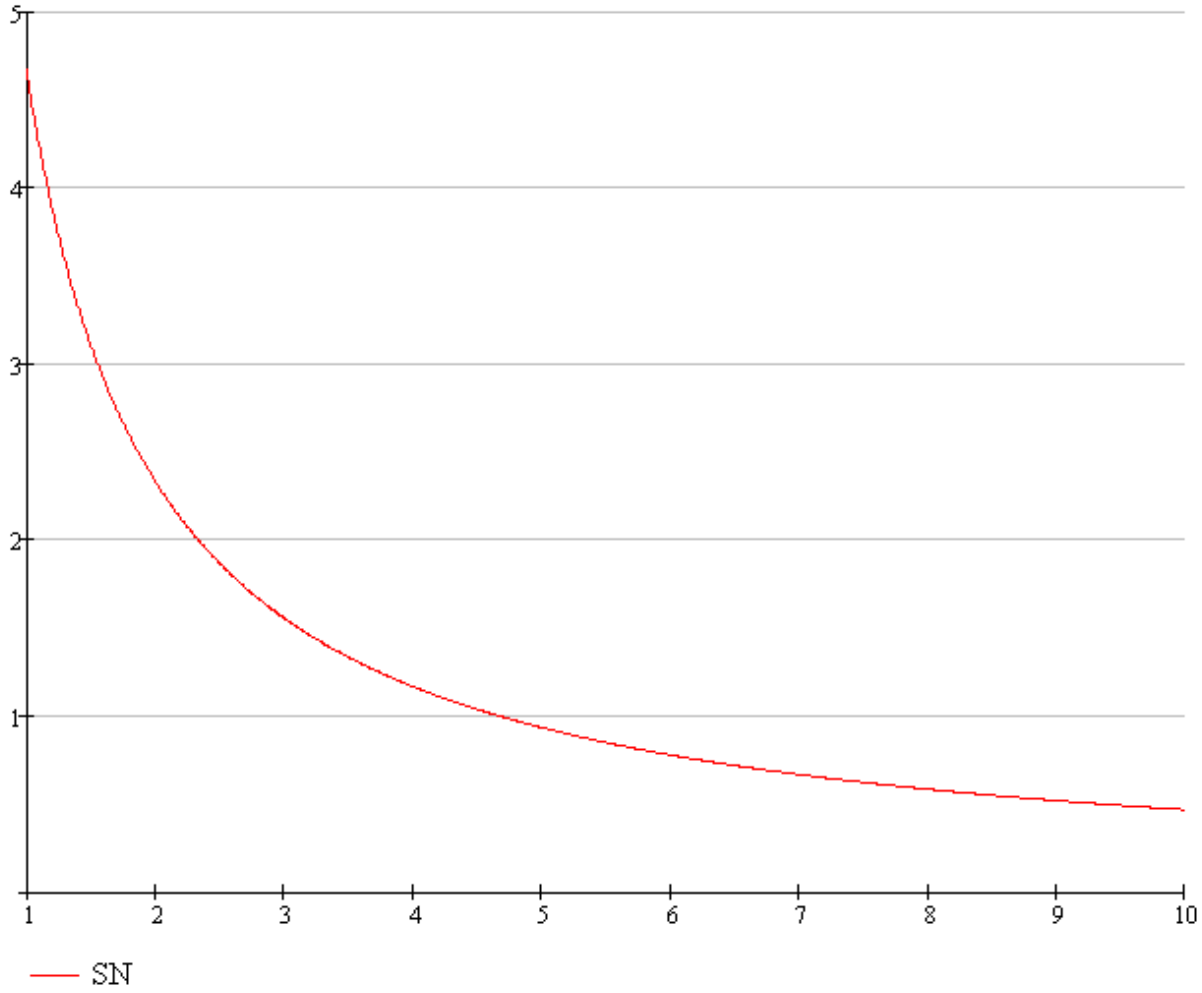


Рис. 5. Зависимость математического ожидания времени выполнения запроса к таблицам архитектурах SN от числа процессоров.

Анализ графиков позволяет сделать несколько выводов:

1. Графики для архитектур SE и SD практически совпали. Это объясняется высокими значениями интенсивностей μ_M и μ_N.

2. При n=2 среднее время выполнения запроса в архитектуре SN в три раза меньше чем для архитектур SE и SD.

4. Для SE и SD при n>2 перегружается дисковая подсистема, и дальнейший рост числа процессоров не имеет смысла. Для архитектуры SN время продолжает уменьшаться с ростом n (здесь нет разделяемых ресурсов). Однако следует иметь в виду, что для снижения времени выполнения запроса с 2 секунд до 1 необходимо увеличить количество процессов с 2 до 4 (см. рис. 4 и рис. 5), а это может оказаться экономически не целесообразным.

Заключение

1. Получены преобразования Лапласа-Стилтьеса (ПЛС) случайного времени выполнения запроса к таблице в параллельной системе баз данных для различных архитектур (SE, SD, SN). Эти преобразования позволяют оценивать не только математические ожидания случайного времени, но моменты более высоких порядков (например, дисперсии).

2. Получены выражения для математических ожиданий времени выполнения запроса для указанных выше вариантов. Рассмотрен практический пример расчета, позволивший сделать ряд нетривиальных выводов.

3. В дальнейшем предполагается использовать аппарат ПЛС для оценки времени выполнения соединения таблиц и аналитических запросов к хранилищу данных, реализованному на основе параллельной системы баз данных и использующему специальные планы соединения таблиц измерений и фактов.

ЛИТЕРАТУРА

1. Соколинский Л. Б., Цымблер М. Л. Лекции по курсу "Параллельные системы баз данных" // pdbs.susu.ru: Практикум по программированию параллельных систем баз данных. Электронный учебный курс. URL. http://pdbs.susu.ru/CourseManual.html (дата обращения: 10.04.2009).
2. Соколинский Л.Б. Обзор архитектур параллельных систем баз данных // Программирование. 2004. № 6. С. 49-63.
3. David Dewitt, Jim Gray. Parallel database systems: the future of high performance database systems // Communications of the ACM. June 1992. Vol. 35, № 6. P.1-26.
4. Григорьев Ю.А., Плутенко А.Д. Теоретические основы анализа процессов доступа к распределённым базам данных. Новосибирск: Наука, 2002. 180 с.
5. Яшков С.Ф. Анализ очередей в ЭВМ. М.: Радио и связь, 1989. 216 с.
6. Авен О.И., Гурин Н.Н., Коган Я.А. Оценка качества и оптимизация вычислительных систем. М.: Наука, 1982. 464 с.
7. TPC-C // www.tpc.org: Transaction Processing Performance Council (TPC). URL. http://www.tpc.org/tpcc/default.asp (дата обращения: 10.04.2009).
8. InfiniBand™ Frequently Asked Questions // www.mellanox.com: Mellanox Technologies. URL. www.mellanox.com/pdf/whitepapers/InfiniBandFAQ_FQ_100.pdf (дата обращения: 26.11.2010).
9. Левин Л. Teradata совершенствует хранилища данных // www.pcweek.ru: PC Week. URL. http://www.pcweek.ru/themes/detail.php?ID=71626 (дата обращения: 26.11.2010)