УДК 623.618.3

Модель информационного обеспечения систем управления реального времени при решении задач с широким спектром входных данных

Дудаков Н.С*., Макаров К.В., Путято С.А.*****

Группа компаний «РТИ»,

ул. 8 Марта,10, стр. 1, Москва, 127083, Россия

**e-mail: nikolay.dudakov@gmail.com*

***e-mail: kmakarov@oaorti.ru*

****e-mail: sputyato@oaorti.ru*

Аннотация

В работе решается задача управления хранением данных с широким спектром характеристик. Предложено использование аппарата теории массового обслуживания для описания процесса обработки и хранения информации. Решается задача оптимизации структуры СУБД при работе с разнородными данными.

Ключевые слова: системы управления базами данных, автоматизированные системы управления, теория массового обслуживания.

Введение

При проектировании современных информационных систем одним из наиболее важных моментов является управление доступом, хранением и использованием информации. С увеличением объемов информации и сложности

вычислений при анализе и обработке данных возрастают требования к системам управления базами данных (СУБД) информационных систем.

Проектирование и исследование баз данных представляет собой динамично развивающуюся область разработки программного обеспечения. Вопросы разработки и проектирования СУБД широко представлены в научной и прикладной литературе, как в классических работах (К. Дж. Дейт, Э. Кодд, Т. Коннелли) [1], так и в материалах по современным реализациям СУБД (*Oracle, DB2, PostgreSQL,* Линтер).

В большинстве современных проектов в качестве универсальной СУБД используется клиент-серверная объектно-реляционная промышленная СУБД *PostgreSQL*. Зачастую, *PostgreSQL* удовлетворяет поставленным при проектировании требованиям, тем не менее, при высокой нагрузке, время выполнения ряда запросов может быть уменьшено на 90% заменой промышленной СУБД на более простую и, в ряде случаев, быструю локальную базу данных (БД). В то же время, возможностей простых и быстрых локальных БД не достаточно для выполнения всех функциональных требований.

Для сравнения производительности баз данных при определенной нагрузке построена зависимость времени выполнения запросов на чтение для промышленной клиент-серверной СУБД, сервер которой расположен локально и удаленно соответственно, а также для движка БД *SQLite* – простого и производительного локального хранилища данных (Рис.1):

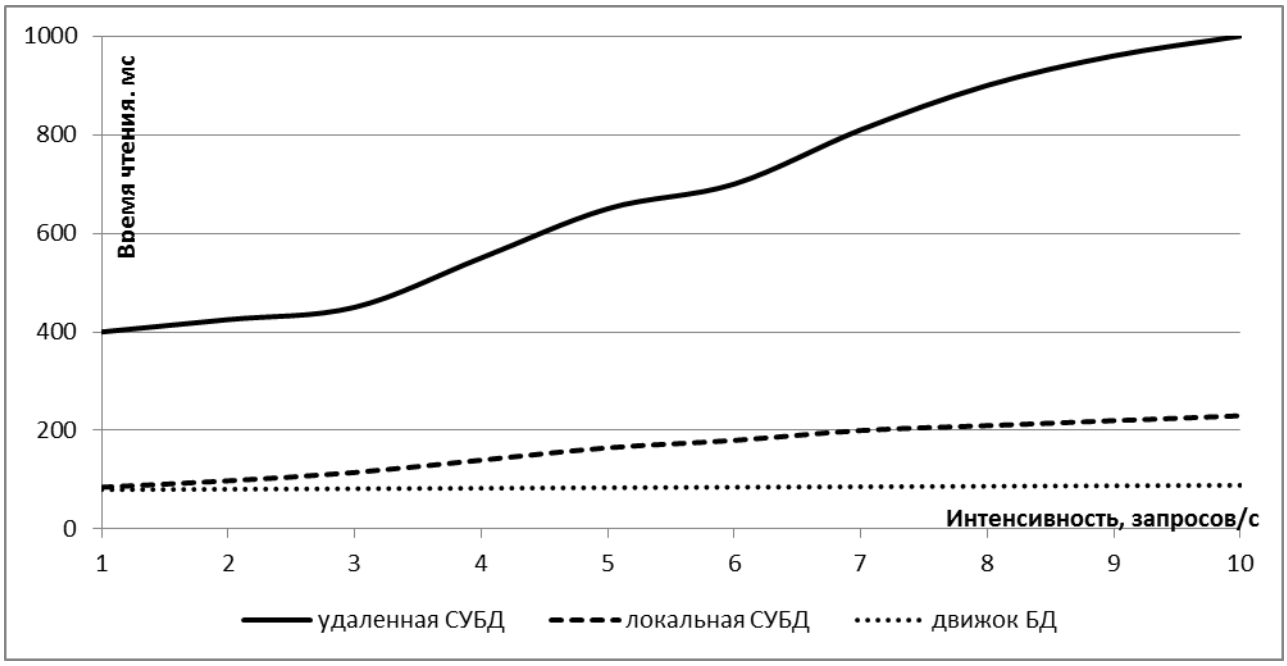

Рис.1 Время чтения данных при повышении интенсивности запросов

Зависимость времени чтения данных при росте интенсивности запросов показывает существенную зависимость времени чтения данных от состояния и загрузки каналов между сервером СУБД (кластером серверов) и рабочим местами пользователей. Также, использование более простой и быстрой локальной БД позволяет сохранить малое время чтения данных при значительном росте нагрузки. В то же время, возможностей простых и быстрых локальных БД не достаточно для выполнения всех функциональных требований.

Оценка эффективности СУБД при решении задач с широким спектром входных данных

Одним из вариантов решения задачи хранения и обработки данных при высокой нагрузке может быть оптимизация структуры СУБД при неизменных программных и аппаратных средствах. При возможности кластеризации обрабатываемых данных по темпам обновления и доступа на малосвязанные

области, актуальным является использование распределенных неоднородных СУБД [2] (РН СУБД): подбор распределения данных, с учетом их характеристик, по частям РН СУБД позволит обрабатывать их с большей эффективностью. Недостатком применения РН СУБД, очевидно, является снижение надежности системы в целом и необходимость поддержки ссылочной целостности всей системы.

Для описания процесса обработки запросов в СУБД в работе предлагается провести математическое моделирование СУБД, для чего использовать аппарат теории массового обслуживания (ТМО). Математическое моделирование распределенных систем широко используется, в частности, при проектировании систем специального назначения [3].

При описании модели, хранимую информацию удобно представлять в виде объектов, объекты группируются в классы данных, каждый объект принадлежит единственному классу.

Для формальной постановки задачи вводятся следующие обозначения:

$n_j \in \boldsymbol{N}, j = \overline{1,N}$ - класс данных;

$\mathrm{m_i} \in \boldsymbol{M}, i = \overline{1,M}$ - компонент РН СУБД.

Исходя из аппарата ТМО, система, состоящая из выбранного хранилища и потока запросов группы классов, рассматривается, как система массового обслуживания (СМО), обрабатывающая случайный поток заявок. СМО рассматривается, как система с бесконечным ожиданием, т.к., отбрасывание устаревших пакетов осуществляется на уровне подключаемых служб и приложений. Поступление информации от большого числа источников сложно предсказуемо, на

основании чего время между поступлениями информации считается случайным. Последовательный разбор и пакетное поступление данных позволяют считать, что в каждый момент времени может произойти только одно событие.

Согласно классическим работам (теорема Пальма, предельная теорема Хинчина) [4], «реальный» поток данных с достаточным приближением описывается, как пуассоновский, число событий k на временном интервале $t>0$ распределено с вероятностью:

$$P(k,t) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}, \quad (1)$$

где $\lambda > 0$ – интенсивность потока событий.

При рассмотрении времени обработки запросов, тип входящей заявки можно считать случайным - дискретной случайной величиной с вероятностью получения заявки класса n_j, определяемой следующим образом:

$$P(n_j) = \lambda_j \Big/ \sum_{l=1}^{N} \lambda_l \quad (2)$$

В соответствии с аддитивностью пуассоновского потока, считается, что в систему поступает один поток с интенсивностью $\lambda = \sum_{l=1}^{N} \lambda_l$ и случайным, дискретно распределенным временем обслуживания b, функция распределения (Ф.Р.) которого B(t) – ступенчатая функция.

Для описания системы предлагается использовать «элемент принадлежности» класса к хранилищу:

$$x_{ij} = \begin{cases} 1, n_j \to m_i \\ 0, n_j \not\to m_i, \end{cases} \quad (3)$$

где обозначение $n_j \to m_i$ определяет обработку объектов класса n_j в компоненте m_i.

Далее, для связанных классов (выполнение запроса к классу n_{j1} требует предварительного выполнения запроса n_{j2}) можно полагать реальное время выполнения заявки:

$$b'_{j1} = b_{j1} + \sum_{i=1}^{M} x_{ij2}\, b_{ij2} \quad (4)$$

Для запросов на чтение, которые в большинстве современных баз данных являются неконкурентными между собой, вводится отдельная величина v_{ij} – время выполнения запроса типа n_j на чтение. Тогда, учитывая одновременную возможность выполнения запросов на чтение, для каждого хранилища рассматривается запрос отдельного типа n_{N+i} с интенсивностью:

$$\lambda_{iN+1} = \sum_{l=1}^{N} \lambda_l x_{il} \quad (5)$$

и временем обработки заявки:

$$b_{iN+1} = \frac{1}{N} \sum_{l=1}^{N} v_l x_{il} \quad (6)$$

Рассматривая описанную модель, искомой величиной является среднее время ожидания заявки:

$$w = \frac{\lambda b^{(2)}}{2(1-\lambda b)}, \tag{7}$$

где $b^{(2)}$ – второй момент времени обслуживания, время обработки заявок – дискретная случайная величина с Ф.Р. *B(t)*:

$$b = b^{(1)} = \sum_{i=0}^{N} b_i P(b_i); \; b^{(2)} = \sum_{i=0}^{N} (b_i{}^2) P(b_i) \tag{8}$$

Соответственно, «реальное» (ожидаемое) время выполнения заявки n_j:

$$\tilde{b}_{ij} = (w_i + b_{ij}) x_{ij} \tag{9}$$

Линейная зависимость величин b_{iN+1} и b'_{j1} от переменных x_{ij} позволяет использовать модель без ограничения общности.

Для оценки точности модели проводилось сравнение времени ожидания выполнения запросов для движка БД SQLite (Рис.2):

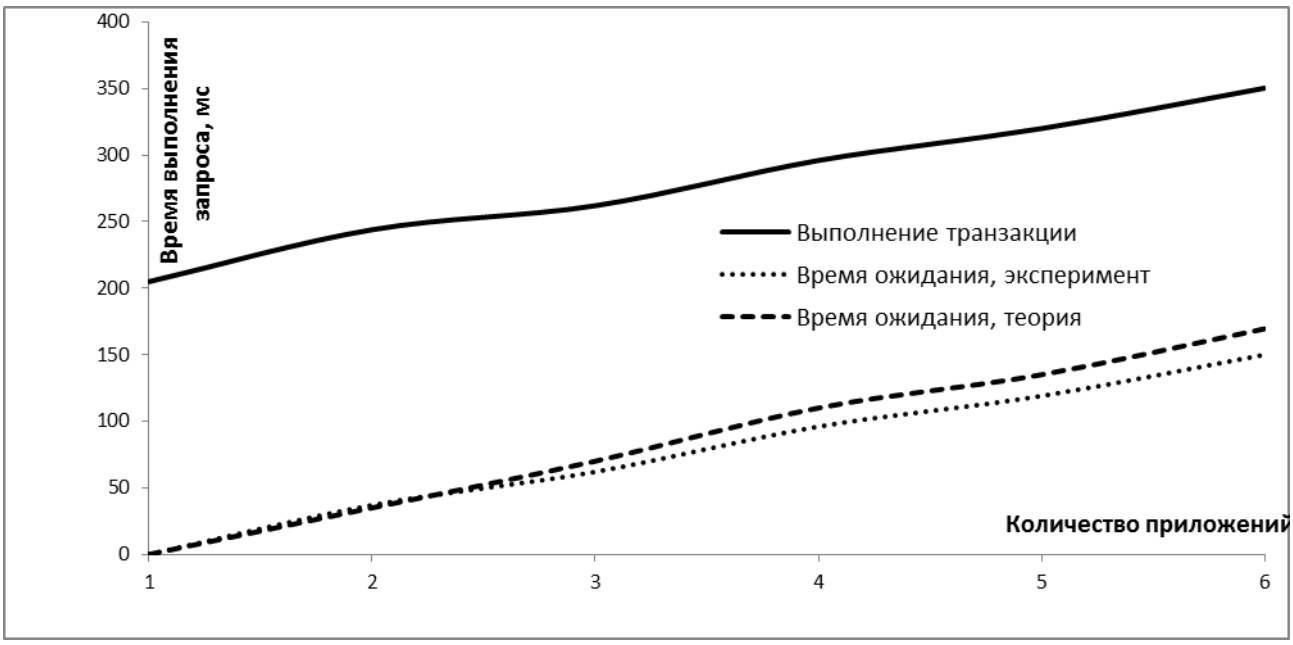

Рис.2 Точность моделирования при повышении интенсивности запросов

Построенные зависимости времени ожидания выполнения запросов при росте интенсивности нагрузки показывают, что модель является адекватной и применимой даже при конкурентном доступе к БД.

Рассматривая распределение $X = \{x_{ij}\}_{i=\overline{1,M},j=\overline{1,N}}$ данных по РН СУБД, критерием эффективности для одного хранилища предлагается считать среднее время ожидания выполнения запроса,

$$J_1(m_i, X) = w_i = \frac{\sum_j b_{ij}^2 \lambda_j x_{ij}}{2(1 - \sum_j b_{ij} \lambda_j x_{ij})}, \quad (10)$$

Для системы хранилищ, критерий построен на основе полного времени пребывания заявки в системе:

$$J(X) = \sum_j \tilde{b}_{ij} = \sum_i^M \sum_j^N (b_{ij} + w_i) x_{ij} \quad (11)$$

Критерий позволяет учесть загрузку всех хранилищ соответственно обрабатываемым наборам классов и динамику поступления входящих заявок.

Оптимальное управление структурой СУБД

Предложенные модель и критерии позволяют количественно оценить эффективность СУБД, при этом возникает задача построения соответствия классов (разбиения данных) и компонентов СУБД.

Задачу поиска оптимального разбиения хранимых данных ставится следующим образом: необходимо найти минимум критерия J (10) на множестве всевозможных распределений $\boldsymbol{X}$ при ограничениях, обеспечивающих отсутствие

накопления очереди при обработке запросов и обработку каждого класса единственным хранилищем:

$$\begin{cases} \forall i = 1,..,M: \sum_{j} (\lambda_j b_{ij} x_{ij}) < 1 \\ \forall j = 1,N, \sum_{i=1}^{M} x_{ij} = 1 \end{cases} \quad (12)$$

Задача оптимизации, поставленная в работе, является задачей псевдо-булева программирования. В качестве метода решения задачи предлагается сведение исходной нелинейной задачи к дробно-линейной путём замены каждого попарного произведения переменных $x_{i1j2} * x_{i3j4}$ на новую переменную $u_{i1j2i3j4} \in \{0,1\}$ с добавлением дополнительных ограничений [5]. Далее, частично линеаризованная задача сводится к широко известной проблеме выполнимости булевых формул (SAT-задача). Суть преобразований при решении задачи оптимизации сводится к получению решения, удовлетворяющего ограничениям (построение приемлемого решения) и итерационному добавлению нового ограничения для отсечения найденного решения.

Также, вариантом решения задачи оптимизации может быть поиск локального оптимального значения критерия (10) при начальном приближении, соответствующем минимуму загрузки компонент СУБД:

$$P(X) = \sum_{i}^{M} \sum_{j}^{N} b_{ij} \lambda_j x_{ij} \rightarrow \min_X. \quad (13)$$

Линейная зависимость выражения (13) от переменных x_{ij} обеспечивает быстрый поиск начального приближения, корреляция загрузки (13) и критерия (10) – быструю сходимость к минимуму критерия.

Сведение задачи оптимизации к задаче выполнимости булевых формул широко распространено вследствие развитого математического аппарата решения SAT-задач. Так, например, задачи кластеризации легко могут быть представлены, как задачи псевдо-булева программирования с соответствующим решением [6]. На настоящий момент применяются два основных типа алгоритмов для решения SAT-задачи: алгоритмы локального поиска, работа которых состоит в модификации некоторого начального приближения до выполняющего набора переменных, и DPLL-алгоритмы (Алгоритм Дэвиса-Патнема-Логемана-Лавленда), которые выполняют поиск в глубину по дереву всевозможных наборов. Программно реализуемые алгоритмы, в основном, сочетают практически более быстрые алгоритмы локального поиска с возможностями DPLL-алгоритмов по упрощению и сокращению размера формул. Несмотря на высокую теоретическую сложность алгоритмов, программы SAT-solvers достаточно эффективны на практике: согласно результатам ежегодно проводимого соревнования, программы за приемлемое время решают задачи, состоящие из порядка 104 переменных и 5·104 ограничений.

Практическая реализация распределения данных с использованием современных реляционных СУБД *PostgreSQL*[7] и БД *SQLite* показывает значительный прирост производительности: по сравнению с использованием только клиент-серверной СУБД *PostgreSQL* (Рис.3):

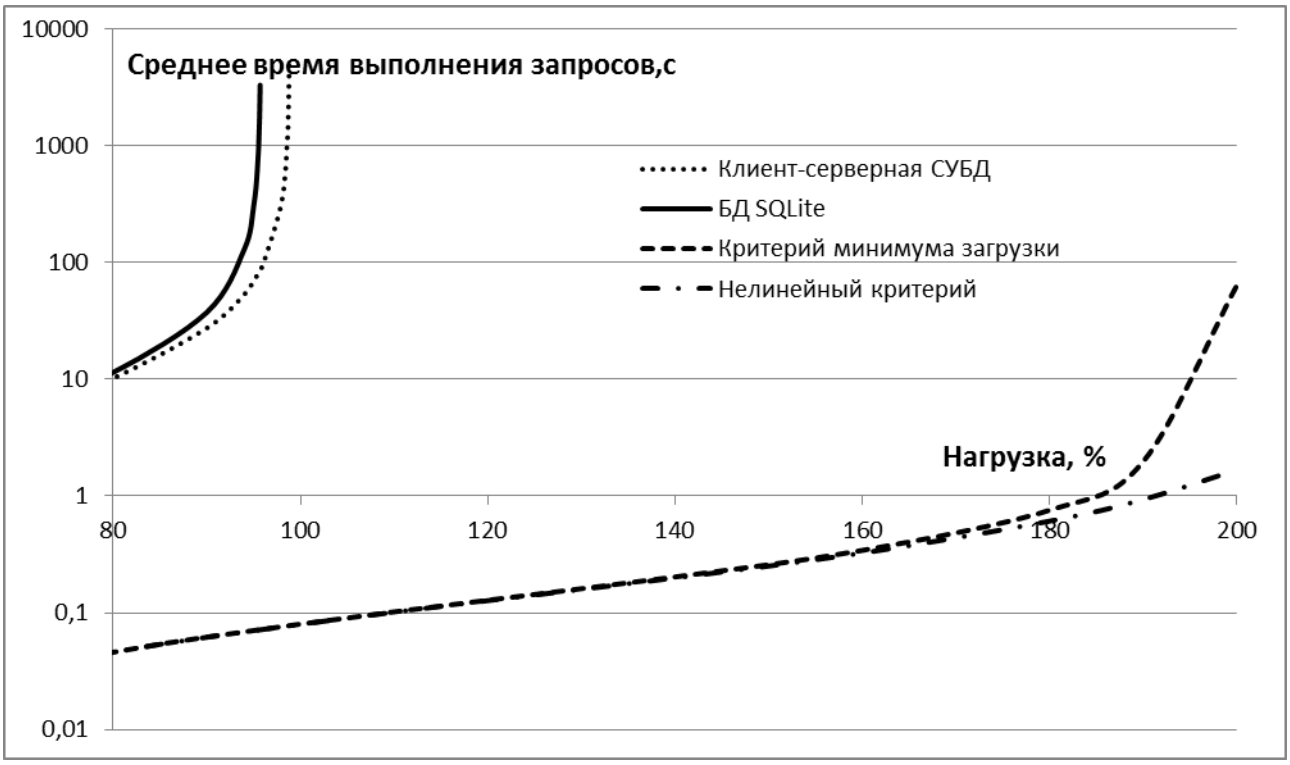

Рис.3 Среднее время выполнения запросов при повышении нагрузки

Построенные зависимости показывают, что оптимальное распределение классов данных значительно уменьшает среднее время выполнения запросов и позволяет вдвое увеличить нагрузку без потери производительности.

Заключение

В работе рассматривалась задача проектирования СУБД для высоконагруженных систем. Вследствие разнородности решаемых задач и существенной зависимости производительности СУБД от характеристик хранимых данных, предлагается использование РН СУБД.

Разработана математическая модель РН СУБД на основе аппарата ТМО, позволяющая количественно охарактеризовать эффективность СУБД с точки зрения производительности и учитывающая неконкурентные и связанные запросы к СУБД.

Согласно полученным экспериментальным данным, модель является точной и адекватной даже при значительной интенсивности запросов.

Поставлена задача оптимизации структуры РН СУБД путем поиска оптимального разбиения хранимых данных по частям РН СУБД. Предложен метод решения задачи оптимизации, как задачи псевдо-булева программирования, на основе сведения к задаче булевой выполнимости.

Пример решения задачи показывает, что разработанная модель, критерии и методы получения оптимального разбиения хранимых данных позволяют получить оценку и обеспечить повышение качества работы СУБД при решении задач с разнородными данными за счет оптимального управления структурой СУБД.

Работа выполнена при поддержке гранта Президента РФ

(№ НШ-6831.2016.8)

Библиографический список

1. Кузнецов С.Д. Основы современных баз данных. – М.: Изд-во «Интернет-университет информационных технологий», 2005. - 488 с.

2. Дейт К. Дж. Введение в системы баз данных. - М.: Вильямс, 2001. 1328 с.

3. Кондрашин М.А., Арсенов О.Ю., Козлов И.В. Применение технологии виртуализации и облачных вычислений при построении сложных распределенных моделирующих систем // Труды МАИ. 2016. № 89. URL: http://trudymai.ru/published.php?ID=73411

4. Клейнрок Л. Теория массового обслуживания - М.: Машиностроение, 1979. - 432 с.

5. Boros E., Peter L. Hammer. Pseudo-Boolean Optimization // Discrete Applied Mathematics. 2002. Vol. 123, pp. 155-225.

6. Калашников В.Б. Сравнительный анализ алгоритмов обнаружения объектов с неизвестной поляризационной матрицей рассеяния методов математического моделирования // Труды МАИ. 2016. № 89. URL: http://trudymai.ru/published.php?ID=73368

7. Муравьев С.К., Дворянкин С.В., Насенков И.И. СУБД: проблема выбора // Открытые системы. СУБД. 2015. № 1. С. 22-24.