УДК 004.657

© 2018 г. **Ю.А. Григорьев,** д-р техн. наук
(Московский государственный технический университет им. Н.Э. Баумана)

ОБ ИСПОЛЬЗОВАНИИ СТОИМОСТНОЙ МОДЕЛИ ДЛЯ ПРОГНОЗИРОВАНИЯ ВРЕМЕНИ ВЫПОЛНЕНИЯ ЗАПРОСОВ К БАЗЕ ДАННЫХ

Проанализирована возможность применения стоимостной модели оптимизатора запросов к базе данных для оценки времени выполнения операторов Select. Доказано, что при использовании откалиброванной стоимостной модели относительная погрешность вычисления оценки времени выполнения запросов стремится к нулю при увеличении объема базы данных даже при неточных значениях кардинальности.

Ключевые слова: стоимостная модель, оптимизатор запросов, прогнозирование времени, погрешность оценки.

DOI: 10.22250/isu.2018.55.3-15

Введение

Предсказание времени выполнения запросов к базе данных всегда было важной задачей. В последнее время эта задача приобрела еще большее значение в контексте предоставления баз данных в качестве сервиса (DaaS) [1]. Поставщик DaaS должен управлять инфраструктурными расходами, а также соглашениями об уровне обслуживания (SLA). Оценки времени выполнения запросов могут быть использованы в процессе управления системой [1]:

1. Контроль доступа. Знание времени выполнения входящего запроса позволяет принимать решение о возможности его обработки [2, 3].

2. Планирование запросов. Знание времени имеет важное значение при планировании задержек и предельных сроков выполнения запросов [4, 5].

3. Мониторинг прогресса. Знание времени выполнения входящего запроса может помочь избежать «запросов-изгоев», которые отправляются с ошибкой и выполняются неоправданно долгое время [6].

4. Калибровка системы. Знание времени выполнения запроса в зависимости от характеристик аппаратных ресурсов может помочь при проектировании и настройке системы [7].

Некоторые работы по прогнозированию времени выполнения запросов [2, 3, 8, 9] были посвящены различным методам машинного обучения, которые рассматривают СУБД как черный ящик и пытаются построить прогностическую модель. Этот шаг к методам машинного обучения черного ящика неявно, а иногда явно мотивирован убеждением, что оценки затрат, полученных оптимизатором запросов, недостаточно хороши для прогнозирования времени их выполнения. Например, в [9] авторы обнаружили, что линейная регрессия, используемая для сопоставления стоимости коммерческих запросов Neoview с фактическим временем, неточна. В [8] пришли к аналогичным неутешительным результатам.

Однако результаты экспериментов, приведенных в [1], свидетельствуют, что если модель оптимизатора предварительно настроена, то точность получаемых оценок времени выполнения запросов часто выше, чем точность оценок, которые получены методами машинного обучения.

В работе [10] показано, что калибровка стоимостной модели и использование в ней точных значений кардинальности (числа записей) таблиц позволяют построить достаточно адекватную прогностическую модель для реального наполнения базы данных (3 ГБ). Однако в этой же работе приведены результаты экспериментов, свидетельствующие, что при том же объеме базы данных и неточных значениях кардинальности (т.е. рассчитанных с использованием стоимостной модели оптимизатора СУБД) ошибка прогнозирования времени выполнения запросов высока.

В данной статье доказывается, что при использовании откалиброванной стоимостной модели относительная погрешность вычисления оценки времени выполнения запросов стремится к нулю при увеличении объема базы данных даже при неточных значениях кардинальности.

Анализ использования стоимостной модели СУБД PostgreSQL для прогнозирования времени выполнения запросов на тестах TPC-H

Ниже приведены свойства, на основе которых модель оценки затрат на выполнение запросов к СУБД следует считать стоимостной [10]:

1. Равномерность распределения: предполагается, что все значения какого-либо атрибута распределены равномерно в заданном интервале.

2. Независимость: значения атрибутов (в той же таблице или в разных таблицах) считаются независимыми.

3. Принцип включения: домены ключей соединения перекрываются так, что ключи из меньшего домена имеют совпадения в более крупном домене.

Наборы данных, которые действительно обладают перечисленными выше свойствами 1 – 3, называются синтетическими.

В этом разделе анализируются результаты экспериментов, которые приведены в работе [1]. Сначала рассмотрим модель затрат PostgreSQL [1].

Оптимизатор PostgreSQL использует вектор из пяти параметров (называемых единицами затрат): C = (cs, cr, ct, ci, co), определяемый следующим образом: 1) cs: стоимость ввода-вывода для последовательного доступа к странице; 2) cr: стоимость ввода-вывода для случайного доступа к странице; 3) ct: стоимость процессора для обработки кортежа; 4) ci: стоимость процессора для обработки кортежа через индексный доступ; 5) co: стоимость процессора для выполнения операции, такой как хэш или агрегация.

Стоимость (CO) SQL-оператора O в плане запроса затем вычисляется линейной комбинацией cs, cr, ct, ci и co:

$$CO = N^T C = ns \cdot cs + nr \cdot cr + nt \cdot ct + ni \cdot ci + no \cdot co. \quad (1)$$

Здесь значения N^T = (ns, nr, nt, ni, no) представляют оценку числа последовательно просматриваемых страниц, числа случайно выбранных страниц и т. д. во время выполнения оператора.

Следовательно, точность CO зависит как от точности C, так и от N. В PostgreSQL cs = 1.0, cr = 4.0, ct = 0.01, ci = 0.005 и co = 0.0025 по умолчанию. Эти единицы затрат были несколько произвольно заданы разработчиками оптимизатора, без учета характеристик системы, на которой выполняется запрос. Использование линейной регрессии для сопоставления полученной оценки стоимости и времени выполнения запроса будет работать только в том случае, если отношения между этими единицами правильны. И неудивительно, что эти коэффициенты по умолчанию были далеко не правильными в исследованных системах [1].

1. Калибровка параметров C. В [1, 11] под каждый настраиваемый коэффициент из C стоимостной модели (или группу коэффициентов) подбирается запрос Q_i. Затем решается система линейных уравнений. Но получаемые значения могут иметь случайный характер. Лучше (1) рассматривать как регрессию и оценивать коэффициенты C методом наименьших квадратов (МНК).

2. Уточнение N. Главная задача при оценке N состоит в том, чтобы уточнить кардинальность (число записей) ввода – вывода для каждого оператора запроса. Для этого часто используется подход [1, 10], основанный на выборке. Выбирается какой-то процент записей из таблицы (или из нескольких таблиц – для соединения) и оценивается селективность p – доля записей, удовлетворяющих условию оператора. Затем кардинальность вычисляется как: $p \cdot |R|$, где $|R|$ – число записей в таблице, участвующей в операции.

В [1] эксперименты проводились на двух разных конфигурациях оборудования:

PC1: 1-core 2.27 GHz Intel CPU и 2GB ОП;

PC2: 8-core 2.40 GHz Intel CPU и 16GB ОП.

Результаты калибровки параметров оптимизатора PostgreSQL представлены в таблице.

Параметр оптимизатора	Откалибровано μ±σ (ms) для PC1	Откалибровано μ±σ (ms) для PC2	Default
seq page cost (cs)	5.53e-2±3.09e-3	5.03e-2±3.82e-3	1.0
rand page cost (cr)	6.50e-2±2.32e-2	4.89e-1±7.44e-2	4.0
cpu tuple cost (ct)	1.67e-4±5.83e-6	1.41e-4±1.35e-5	0.01
cpu index tuple cost (ci)	3.41e-5±2.30e-5	3.34e-5±3.85e-5	0.005
cpu operator cost (co)	1.12e-4±1.30e-6	7.10e-5±1.52e-5	0.0025

Из таблицы видно, что действительные соотношения параметров оптимизатора определяются значениями: для PC1 – 1:1,2:0,003:0,0006:0,002; для PC2 – 1:10:0,003:0,0007:0,0014. Они существенно отличаются от значений по умолчанию (столбец Default).

Исходный генератор базы данных TPC-H использует однородные распределения. Чтобы проверить надежность различных подходов к разным распределениям данных, в [1] использовался перекошенный генератор базы данных TPC-H [12]. Этот генератор заполняет базу данных TPC-H, используя дистрибутив Zipf. Распределение имеет параметр z, который контролирует степень асимметрии. z = 0 генерирует равномерное распределение, а при увеличении z данные становятся все более искаженными. Перекошенные базы данных создавались с использованием z = 1. В [1] использовались запросы из теста TPC-H. Случайным образом выбирались 10 запросов из каждого из 21 шаблонов Q1-Q22 (Q15 был исключен из рассмотрения, так как представление в этом запросе не поддерживалось в эксперименте), и каждый запрос выполнялся 5 раз. Между каждым прогоном каждого запроса очищалась как файловая система, так и буферы БД.

На рис. 1 – 4, приведенных в [1], показаны средние относительные ошибки расчетов времени выполнения запросов с применением стоимостной модели PostgreSQL. Везде использовалась калибровка параметров C. На рисунках приняты следующие обозначения: E_t – используется истинная кардинальность (число записей, удовлетворяющих условию операции запроса); E_0 – используется кардинальность, рассчитанная оптимизатором без улучшения (без дополнительной выборки записей из БД); E_s^f – с улучшенной оценкой кардинальности (для 100% выборки записей из таблиц, участвующих в запросе); E_{SVM}, E_{REP} – используются методы машинного обучения SVM и REP для предсказания времени выполнения запросов; E_0^{LR} – использование регрессии стоимость-время для исходного оптимизатора (без калибровки параметров C).

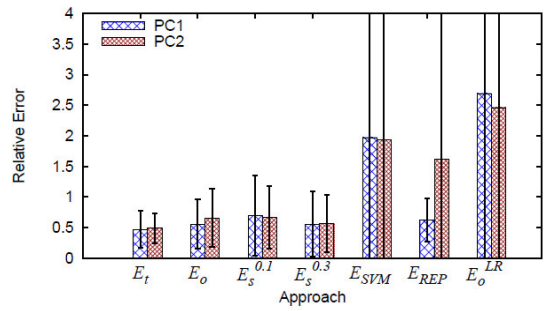

Рис. 1. Равномерное распределение TPC-H, 1GB database, 21 шаблон.

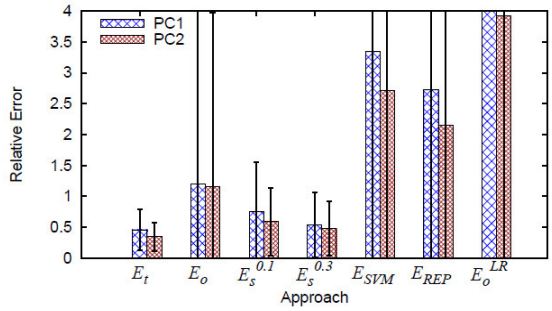

Рис. 2. Неравномерное распределение (Skewed) TPC-H, 1GB database, 21 шаблон.

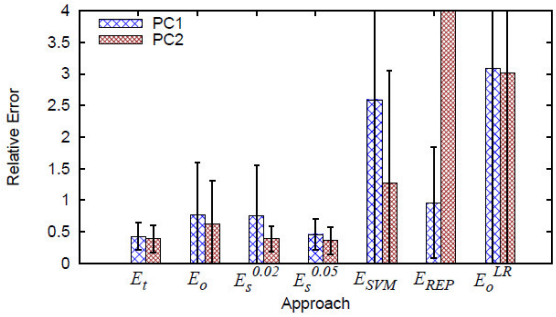

Рис. 3. Равномерное распределение TPC-H, 10 GB database, 21 шаблон.

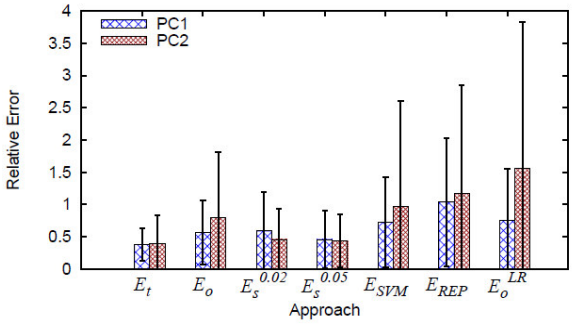

Рис. 4. Неравномерное распределение (Skewed) TPC-H, 10 GB database, 21 шаблон.

Относительная ошибка вычисляется на основе среднего времени выполнения запросов:

$$\mathrm{Re}lative\ \ Error = \frac{1}{M}\sum_{i=1}^{M}\frac{\left|T_i^{pred} - T_i^{act}\right|}{T_i^{act}}, \qquad (2)$$

где M – количество тестовых запросов; T_i^{pred} и T_i^{act} – соответственно предсказанное (по модели) и фактическое время выполнения тестового запроса i.

Точность прогнозирования на основе стоимостной модели сравнивалась [1] с несколькими современными решениями на основе машинного обучения: моделирование на уровне плана с помощью SVM [8] и моделирование на уровне плана с деревьями REP [3]. Один из K шаблонов TPC-H оставался для предсказания (как в [8]), а другие K-1 шаблоны были использованы для создания запросов, с целью обучения прогнозирующих моделей.

По этому разделу сделаем следующие выводы:

1. Калибровка модели и использование точной кардинальности (число записей, удовлетворяющих условию) дают наилучший результат прогнозирования времени выполнения запроса (стратегия E_t). Погрешность моделирования не превышает 50% (см. рис. 1 – 4).

2. Ошибка прогнозирования для E_0 близка к ошибке для E_t (см. рис. 1). Улучшение оценки кардинальности (E_t) не помогает, так как данные независимы

и распределены равномерно. Для неравномерного распределения (см. рис. 2) стратегия E_s^f лучше E_0.

3. При увеличении объема базы данных (см. рис. 3 и 4) только калибровка стоимостной модели (стратегия E_0) дает среднюю ошибку оценки времени выполнения запросов на уровне 50-75%, приемлемую на этапе проектирования, когда высока неопределенность исходных данных. Это справедливо для равномерного и неравномерного распределения значений атрибутов базы данных. Причем оценка ошибки близка к оценкам ошибок моделирования для стратегий E_t и E_s^f. Только для стратегии E_0 дисперсия ошибки выше.

4. Остальные стратегии (E_{SVM}, E_{REP}, E_0^{LR}) оценки времени выполнения запросов хуже предыдущих (E_t, E_0, E_s^f) (см. рис. 1 – 4).

Как указано в [1], одна из возможных причин большой ошибки прогнозирования с использованием стратегий E_{SVM}, E_{REP} заключается в следующем: большинство методов машинного обучения предполагает, что тестовые запросы должны быть похожи на запросы, используемые при обучении модели. Более конкретно: векторы признаков тестовых запросов должны быть близки к векторам признаков обучающих запросов с точки зрения расстояния в пространстве признаков. К сожалению, это предположение неверно для динамических рабочих нагрузок. Шаблоны могут быть сгруппированы в несколько кластеров [1]. Примерно половина шаблонов (из 21) попадает в один кластер, и каждый из оставшихся шаблонов обычно образует одноэлементный кластер.

Анализ регрессии «стоимость-время» выполнения запросов на реальной базе данных

Здесь используются результаты экспериментов, которые приведены в [10].

Многие исследовательские работы по обработке и оптимизации запросов применяют стандартные эталонные тесты – такие как TPC-H, TPC-DS или тесты Star Schema Benchmark (SSB). Чтобы иметь возможность легко масштабировать данные, используемые тестами, генераторы создают синтетические наборы данных. Напротив, наборы данных реального мира полны корреляций и неравномерных распределений данных, что значительно усложняет оценку кардинальности.

Поэтому вместо синтетического набора данных в [10] выбрали базу данных Интернет-фильмов (IMDB). Она содержит много информации о фильмах и связанных с ними фактах, об актерах, директорах, производственных компаниях и т.д. База данных позволяет отвечать на запросы типа «Какие актеры играют в фильмах, выпущенных в период с 2000 по 2005 г. с рейтингами выше 8?». Как и в большинстве реальных наборах данных, IMDB полна корреляций и неравномерных распределений данных и, следовательно, гораздо сложнее, чем большинство

синтетических наборов данных. Используемый снимок создан в мае 2013 г. и занимает 3,6 ГБ при экспорте в файлы CSV. Две крупнейшие таблицы, информация о ролях и информация о фильмах, содержат 36МБ и 15МБ строк соответственно. Набор данных IMDB был загружен [10] в 5 реляционных систем баз данных PostgreSQL, HyPer и три коммерческих СУБД (DBMS A, DBMS B, DBMS C).

На основе базы данных IMDB в [10] были реализованы аналитические SQL-запросы. Поскольку в работе [10] рассматриваются упорядоченные соединения (это самая важная задача оптимизации запросов), запросы включают от 3 до 16 соединений таблиц, в среднем по 8 соединений для каждого запроса. Например, запрос 13d, который находит рейтинги и даты выпуска для всех фильмов, выпущенных американскими компаниями, является типичным и имеет следующий вид:

```
SELECT cn.name, mi.info, miidx.info
    FROM company_name cn, company_type ct, info_type it, info_type it2,
title t, kind_type kt, movie_companies mc, movie_info mi, movie_info_idx miidx
WHERE cn.country_code =’[us]’ AND ct.kind = ’production companies’
AND it.info = ’rating’ AND it2.info = ’release dates’ AND kt.kind = ’movie’
AND ... -- (11 join predicates).
```

Набор запросов состоит из 33 структур запросов, каждый из которых имеет 2-6 вариантов, которые различаются только селективностью, в результате в общей сложности использовались 113 запросов. В зависимости от селективности предикатов базовой таблицы варианты одной и той же структуры запроса имеют разные оптимальные планы запросов, которые дают значительно отличающееся (иногда на порядок) время выполнения.

Кроме того, некоторые запросы имеют более сложные предикаты выбора, чем приведенный выше (например, дизъюнкции или поиск подстроки с использованием LIKE).

Все эксперименты в [10] по оценке производительности были выполнены на сервере с двумя процессорами Intel Xeon X5570 (2,9 ГГц) и в общей сложности с 8 ядрами, работающими под управлением PostgreSQL 9.4 на Linux. PostgreSQL не распараллеливает запросы, во время обработки запросов используется только одно ядро. Система имеет 64 ГБ ОП, поэтому вся база данных IMDB полностью кэшируется в ОП.

Результаты промежуточной обработки запросов (например, хэш-таблицы) также легко вписываются в ОП, если не выбран очень плохой план с чрезвычайно большими промежуточными результатами.

Корреляция между стоимостью и временем выполнения запросов в PostgreSQL показана на рис. 5. Значения времени (runtime) и стоимости (cost) приведены в логарифмическом масштабе.

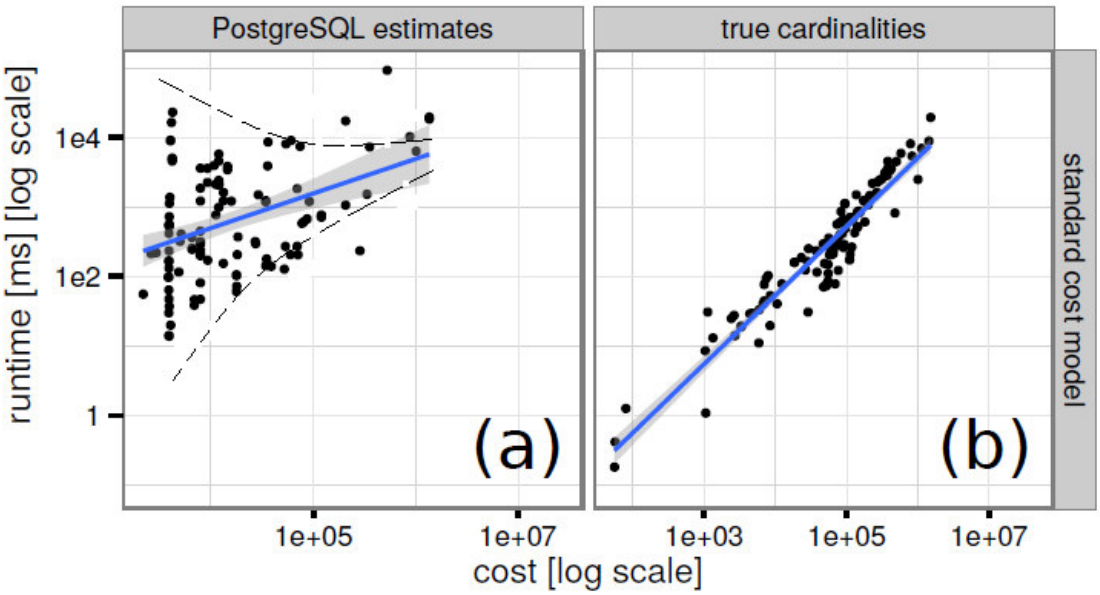

Рис. 5. Прогнозируемая стоимость и время выполнения запросов для стандартной стоимостной модели PostgreSQL [10].

Рассмотрим сначала рис. 5b. Как видно, зависимость времени (T) от стоимости (CO) хорошо описывается линейной регрессией. Действительно из графика имеем

$$\log T = k + \log CO, \qquad (3)$$

где k – некоторая константа.

Отсюда получим

$$T = 10^{k} CO = K \cdot CO. \qquad (4)$$

Средняя ошибка прогнозирования времени t по регрессионной зависимости T от стоимости CO равна 38% (см. рис. 5b). Учитывая выражение для оценки стоимости (1), имеем

$$T = K \cdot CO = K \cdot N^{T} C = N^{T}(K \cdot C) = N^{T} \cdot C_{K}, \qquad (5)$$

где C_K – вектор откалиброванных коэффициентов C стоимостной модели.

Из рис. 5b и выражения (5) следует, что калибровка коэффициентов модели и задание точных значений кардинальности исходных и промежуточных таблиц (числа записей), используемых для оценки элементов вектора N^T, позволяют построить хорошую прогностическую стоимостную модель оценки времени выполнения запросов для реального наполнения базы данных.

Обратимся теперь к рис. 5a. Регрессионная зависимость близка к прямой, представленной выражением (3) (на рис. 5a серым цветом подсвечена область отклонения регрессионной прямой). Поэтому для нее справедлива формула (5). Но так как оценка кардинальности, выполненная стандартным оптимизатором PostgreSQL, не является точной, то и оценка элементов вектора N^T также не точна. Поэтому даже после калибровки стоимостной модели имеет место большое отклонение времени выполнения запросов от регрессионной прямой (см. рис. 5a). В

то же время с увеличением стоимости запроса (т.е. с увеличением кардинальности таблиц) разброс времени выполнения запросов относительно регрессионной прямой уменьшается (см. конус на рис. 5а, обозначенный пунктирными кривыми). Это согласуется с результатами экспериментов, выполненными в [1]. Приведенные в работе [1] данные свидетельствуют, что при увеличении объема базы данных только калибровка стоимостной модели (стратегия E_0) дает приемлемую среднюю ошибку оценки времени выполнения запросов (на уровне 50-75%) даже при использовании стандартного оптимизатора PostgreSQL (см. рис. 3 и 4).

Из рис. 5а, 3 и 4 следует, что калибровка параметров модели С позволяет построить хорошую прогностическую стоимостную модель оценки времени выполнения запросов для баз данных с большим наполнением даже при неточных значениях кардинальности исходных и промежуточных таблиц. Ниже приведено теоретическое обоснование этого утверждения.

Представим время t в виде случайной величины:

$$t = K \cdot CO + \xi, \qquad (6)$$

где $M(\xi) = 0$, т.е. математическое ожидание t равно

$$M(t) = T = K \cdot CO. \qquad (7)$$

Оно совпадает с регрессионной прямой на рис. 5а (см. также формулу (4)).

Случайное время выполнения запроса представим в виде следующего выражения:

$$t = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{k=1}^{|R_j|} \xi_{ijk}, \qquad (8)$$

ξ_{ijk} – случайное время обработки k-й записи j-й таблицы при выполнении i-й операции; $|R_j|$ – число записей в j-й таблице. Некоторые ξ_{ijk} являются зависимыми.

В (8) перечисляются таблицы (j = 1,..., J), которые используются хотя бы в одном запросе из выборки: исходные таблицы, декартовы произведения таблиц, участвующих в операции соединения; перечисляются операции (i = 1,..., I), которые выполняются хотя бы в одном запросе из выборки: селекции, соединения хешированием, соединения с использованием индексов (операции генерируются оптимизатором запросов).

Функция распределения случайной величины ξ_{ijk} определяется следующим образом:

$\xi_{ijk} = t_{ijk}$ с вероятностью p_{ijk} и $\xi_{ijk} = 0$ с вероятностью $1\text{-}p_{ijk}$. (9)

По выборке запросов, представленных в виде точек на рис. 5а, оцениваются вероятности p_{ijk} и измеряется время t_{ijk}.

Воспользуемся теперь теоремой Ляпунова [13].

Теорема Ляпунова. Если для последовательности взаимно независимых случайных величин $\{\xi_j\}$ можно подобрать такое $\delta > 0$, что при $n \to \infty$

$$\frac{1}{\mathrm{B_n^{2+\delta}}}\sum_{j=1}^{n}M\left|\xi_j-a_j\right|^{2+\delta}\to 0,\ \mathrm{a}_j=M(\xi_j),\ \mathrm{B_n^2}=D(\sum_{j=1}^{n}\xi_j), \tag{10}$$

то при $n \to \infty$ равномерно по x

$$P\{\frac{1}{\mathrm{B_n}}\sum_{j=1}^{n}M\left|\xi_j-a_j\right|<x\}\to\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}e^{-z^2/2}dz\,. \tag{11}$$

Но некоторые ξ_{ijk} в (8) являются зависимыми величинами. Например, если запись выбрана из исходной таблицы (операция селекции), то она участвует при хешировании (операция соединения).

В [14] определяется максимальный коэффициент корреляции $\rho(n)$ между прошлым $\{\xi_j,\ j \le 0\}$ и будущим $\{\xi_j,\ j \ge n > 0\}$ последовательности $\{\xi_j\}$. Для стационарной последовательности $\{\xi_j\}$ доказывается, что если $\rho(n) \to 0$, то распределение суммы ξ_j стремится к нормальному закону.

Количество слагаемых в (8) велико, ξ_{ijk} можно перенумеровать так, чтобы коррелированные случайные величины размещались в начале последовательности.

Докажем теперь, что распределение вероятностей (9) удовлетворяет условию (10) теоремы Ляпунова.

Положим $\delta = 1$. Тогда

$$\mathrm{r_{3n}}=\sum^{n}M\left|\xi_{ijk}-a_{ijk}\right|^3<C_1n\,, \tag{12}$$

$$\mathrm{r_{2n}^3}=(\sum^{n}M(\xi_{ijk}-a_{ijk})^2)^{3/2}>C_2n^{3/2}\,, \tag{13}$$

где C_1, C_2 – некоторые константы. Эти константы существуют, так как можно подобрать величины A_1, A_2 такие, что для $\forall i,j,k$ $A_1 < t_{ijk} < A_2$ (см. распределение (9)).

$$\frac{\mathrm{r_{3n}}}{\mathrm{r_{2n}^3}}<\frac{C_1n}{C_2n^{3/2}}\to 0 \text{ при } n\to\infty. \tag{14}$$

Таким образом, условие (10) доказано и функция распределения суммы (8) стремится к нормальному закону.

Из (8) получим математическое ожидание и дисперсию t:

$$\mathrm{T=K\cdot CO=M(t)}=\sum_{\mathrm{i=1}}^{\mathrm{I}}\sum_{\mathrm{j=1}}^{\mathrm{J}}\sum_{\mathrm{k=1}}^{|\mathrm{R_j}|}p_{ijk}t_{ijk}\,, \tag{15}$$

$$\mathrm{D(t)}=\sum_{\mathrm{i=1}}^{\mathrm{I}}\sum_{\mathrm{j=1}}^{\mathrm{J}}\sum_{\mathrm{k=1}}^{|\mathrm{R_j}|}p_{ijk}(1-p_{ijk})t_{ijk}^2\,. \tag{16}$$

Тогда относительная погрешность расчета времени выполнения запроса с использованием откалиброванной стоимостной модели равна

$$\delta = \frac{k_\alpha \sqrt{D(t)}}{M(t)} = \frac{k_\alpha \sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{|R_j|} p_{ijk}(1-p_{ijk})t_{ijk}^2}}{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{|R_j|} p_{ijk}t_{ijk}}, \quad (17)$$

где α – уровень надежности оценки (17); k_α – α-квантиль: 0,95-квантиль = 1,645, 0,99-квантиль = 2,326 и 0,999-квантиль = 3,090.

Далее предположим, что распределения (9) не зависят от k (номера строки таблицы):

$$t_{ijk} = t_{ij},\ p_{ijk} = p_{ij}\ . \quad (18)$$

Подставив (18) в (17) и изменив порядок суммирования, получим:

$$\delta = \frac{k_\alpha \sqrt{\sum_{j=1}^{J}|R_j|\sum_{i=1}^{I} p_{ij}(1-p_{ij})t_{ij}^2}}{\sum_{j=1}^{J}|R_j|\sum_{i=1}^{I} p_{ij}t_{ij}} = \frac{k_\alpha \sqrt{\sum_{j=1}^{J}|R_j|\cdot\eta_{2j}}}{\sum_{j=1}^{J}|R_j|\cdot\eta_{1j}}. \quad (19)$$

Введем фактор наполнения SF исходных таблиц:

$$|R_j| = SF|R_{1j}|. \quad (20)$$

где $|R_{1j}|$ – число записей в таблице R_j с фактором наполнения SF = 1.

Выражение (20) справедливо и для промежуточных таблиц, полученных в результате соединения. Число записей в соединении некоторых таблиц R_m и R_n можно оценить по формуле [10]:

$$|R_j| = \frac{|R_m||R_n|}{\max(I(R_m,a), I(R_n,a))}, \quad (21)$$

где $I(R_m,a)$, $I(R_n,a)$ – мощности атрибута соединения ‘a’ в таблицах R_m и R_n (число элементов в доменах).

Если соединение выполняется по первичному и внешнему ключу, то с увеличением числа записей в таблицах R_m и R_n растут и мощности атрибутов соединения ‘a’ в этих таблицах, т.е.

$$\frac{SF|R_m| \cdot SF|R_n|}{\max(SF \cdot I(R_m,a), SF \cdot I(R_n,a))} = SF \cdot |R_j|. \quad (22)$$

Тогда (19) можно переписать в следующем виде:

$$\delta = \frac{k_\alpha \sqrt{SF \cdot \sum_{j=1}^{J}|R_{1j}|\eta_{2j}}}{SF \cdot \sum_{j=1}^{J}|R_{1j}|\eta_{1j}} = \frac{k_\alpha H}{B\sqrt{SF}}. \quad (23)$$

С ростом SF относительная ошибка расчета по стоимостной модели стремится к 0.

Для уровня надежности α случайное время t выполнения запроса находится в следующем интервале

$$\mathrm{T} \pm \Delta = SF \cdot \mathrm{B} \pm \sqrt{SF} \cdot \mathrm{k}_\alpha \mathrm{H} = SF \cdot B(1 \pm \frac{\mathrm{k}_\alpha \mathrm{H}}{\mathrm{B}\sqrt{\mathrm{SF}}}). \quad (24)$$

Прологарифмируем (24):

$$\log(\mathrm{T} \pm \Delta) = \log T + \log(1 \pm \frac{\mathrm{k}_\alpha \mathrm{H}\sqrt{\mathrm{SF}}}{\mathrm{T}}). \quad (25)$$

Выразим (25) относительно стоимости CO (см. (3)):

$$\log(\mathrm{T} \pm \Delta) = \mathrm{k} + \log CO + \log(1 \pm \frac{\mathrm{k}_\alpha \mathrm{H}\sqrt{\mathrm{SF}}}{\mathrm{K} \cdot CO}), \quad (26)$$

$$\mathrm{H}\sqrt{\mathrm{SF}} = \sqrt{SF \sum_{\mathrm{j}=1}^{\mathrm{J}} \left|\mathrm{R}_\mathrm{j}\right| \sum_{\mathrm{i}=1}^{\mathrm{I}} p_{ij}(1-p_{ij}) t_{ij}^2} < \sqrt{SF \sum_{\mathrm{j}=1}^{\mathrm{J}} \left|\mathrm{R}_\mathrm{j}\right| \sum_{\mathrm{i}=1}^{\mathrm{I}} p_{ij} t_{ij}^2} < \quad (27)$$
$$< \sqrt{TA_2} = \sqrt{K \cdot CO \cdot A_2}.$$

Отсюда для достаточно больших CO получим

$$\log(\mathrm{T} + \Delta) < \mathrm{k} + \log CO + \log(1 + \frac{\mathrm{k}_\alpha \sqrt{\mathrm{A}_2}}{\sqrt{\mathrm{K} \cdot CO}}), \quad (28)$$

$$\log(\mathrm{T} - \Delta) > \mathrm{k} + \log CO + \log\left(1 - \frac{\mathrm{k}_\alpha \sqrt{\mathrm{A}_2}}{\sqrt{\mathrm{K} \cdot CO}}\right).$$

Это объясняет пунктирные кривые на рис. 5а (там графики приведены в логарифмическом масштабе). С увеличением стоимости CO обе кривые стремятся к регрессионной прямой $\log \mathrm{T} = \mathrm{k} + \log \mathrm{CO}$.

Заключение

Выполнен анализ использования стоимостной модели СУБД PostgreSQL для прогнозирования времени выполнения запросов на тестах TPC-H, а также на реальной базе Интернет-фильмов (IMDB). Доказано, что при использовании откалиброванной стоимостной модели, т.е. регрессии «стоимость-время» относительная погрешность вычисления оценки времени выполнения запросов стремится к нулю при увеличении объема базы данных. Это справедливо даже при неточных значениях кардинальности исходных и промежуточных таблиц и для любого уровня надежности оценки (см. выражения (23), (28)).

ЛИТЕРАТУРА

1. *Wu W. et al.* Predicting query execution time: Are optimizer cost models really unusable? //Data Engineering (ICDE), 2013 IEEE 29th International Conference. – IEEE. – 2013. – P. 1081-1092.

2. *Tozer S., Brecht T., Aboulnaga A.* Q-Cop: Avoiding bad query mixes to minimize client timeouts under heavy loads //Data Engineering (ICDE), 2010 IEEE 26th International Conference. – IEEE. – 2010. – P. 397-408.
3. *Xiong P. et al.* ActiveSLA: a profit-oriented admission control framework for database-as-a-service providers //Proceedings of the 2nd ACM Symposium on Cloud Computing. – ACM. – 2011.
4. *Chi Y., Moon H. J., Hacigümüş H.* iCBS: incremental cost-based scheduling under piecewise linear SLAs //Proceedings of the VLDB Endowment. – 2011. – Т. 4, №. 9. – P. 563-574.
5. Guirguis S. et al. Adaptive scheduling of web transactions //Data Engineering, 2009. ICDE'09. IEEE 25th International Conference. – IEEE. – 2009. – P. 357-368.
6. *Mishra C., Koudas N.* The design of a query monitoring system //ACM Transactions on Database Systems (TODS). – 2009. – Vol. 34. – №. 1.
7. *Wasserman T. J. et al.* Developing a characterization of business intelligence workloads for sizing new database systems //Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP. – ACM. – 2004. – P. 7-13.
8. *Akdere M. et al.* Learning-based query performance modeling and prediction //Data Engineering (ICDE), 2012 IEEE 28th International Conference. – IEEE. – 2012. – P. 390-401.
9. *Ganapathi A. et al.* Predicting multiple metrics for queries: Better decisions enabled by machine learning // Data Engineering, 2009. ICDE'09. IEEE 25th International Conference. – IEEE. – 2009. – P. 592-603.
10. *Leis V. et al.* How good are query optimizers, really? //Proceedings of the VLDB Endowment. – 2015. – Т. 9, №. 3. – P. 204-215.
11. *Soror A.A. et al.* Automatic virtual machine configuration for database workloads //ACM Transactions on Database Systems (TODS). – 2010. – Vol. 35, №. 1. – P. 7.1-7.47.
12. Program for TPC-H Data Generation with Skew: https://www.microsoft.com/en-us/download/details.aspx?id=52430.
13. *Гнеденко Б.В.* Курс теории вероятностей. – М.: Наука. Гл. ред. физ.-мат. лит., 1988.
14. *Ibragimov I.A.* A note on the central limit theorems for dependent random variables // Theory of Probability & Its Applications. – 1975. – Vol. 20, №. 1. – P. 135-141.

E-mail:
Григорьев Юрий Александрович – grigorev@bmstu.ru