УДК 004.657

Анализ времени выполнения запроса в параллельном колоночном хранилище данных

© Ю.А. Григорьев, Е.Ю. Ермаков

МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

Проанализирован специфичный для параллельных колоночных хранилищ данных план запроса со скрытым соединением. Приведено преобразование Лапласа — Стилтьеса времени обработки запроса с подобным планом в параллельном колоночном хранилище данных. Выполнено сравнение среднего времени выполнения запроса со скрытым соединением и пересечением NLJ.

***Ключевые слова**: колоночное хранилище данных, колоночные базы данных, параллельные базы данных, преобразование Лапласа — Стилтьеса, скрытое соединение.*

Введение. Являясь одними из наиболее значимых элементов ИТ-инфраструктуры предприятия, базы данных (БД) консолидируют информацию, необходимую для создания достоверных аналитических и управленческих отчетов. Они являются одними из крупнейших источников информации для современных аналитиков и, по оценке Gartner [1], в ближайшей перспективе останутся ключевым компонентом ИТ-инфраструктуры предприятий.

При оценке характеристик производительности на этапе проектирования БД необходимо учитывать особенности предметной области. Результаты исследований [2] показывают, что при расчете времени реакции информационной системы надо учитывать параметры приложений: алгоритмы, запросы к БД и т.д. Время обработки этих запросов достаточно велико, его доля в общем времени выполнения прикладных программ превышает 90 %.

Методы анализа временны́х характеристик для параллельных строчных БД (Oracle, MS SQL Server и т. д.), учитывающих специфику запросов к базе данных, уже разработаны и представлены в работах [2–6]. Но в настоящее время внедряются новые системы управления БД с иной организацией хранения данных, получившие название параллельных колоночных БД (ПКБД) [7–9]. Они впервые были внедрены при разработке больших БД, используемых при поддержке принятия решения, в частности, в аналитических расчетах, и сразу же дали хорошие результаты: почти 200-кратное сокращение объема ввода-вывода по сравнению с аналогичными строчными БД и значительное уменьшение времени выполнения запросов [8]. Это достигается за счет того, что из БД читаются только атрибуты (столбцы), участвующие в запросе, а также применяются эффективные методы сжатия столбцов [10].

Однако проектирование систем на основе колоночных систем управления БД ведется на интуитивном уровне, а кроме того, не существует математических методов, позволяющих учитывать специфику сложных запросов к хранилищу данных, которые используются в процессе принятия решений. Поэтому разработка теоретических методов, дающих возможность на этапе проектирования прогнозировать время работы параллельного колоночного хранилища данных (ПКХД) с учетом специфики предметной области, является в настоящее время актуальной.

В исследовательской работе, проводимой в МГТУ им. Н.Э. Баумана, указанная задача решается путем разработки моделей оценки времени выполнения запроса к ПКХД, учитывающих особенности колоночного хранения данных, состав и параметры выполнения запросов, структуру и наполнение хранилища, механизм распределения таблиц по процессорам системы, параллелизм выполнения запросов в узлах, режимы работы системы, структуру сложного многопроцессорного аппаратно-программного комплекса.

В статье рассмотрено специфическое для параллельных колоночных хранилищ данных скрытое соединение и получена оценка времени выполнения запроса к хранилищу на основе математических методов, предложенных авторами в статьях [11–14] с учетом особенностей выполнения запросов к колоночным БД.

Организация работы колоночной системы БД. Под строчным хранением данных обычно понимают физическое хранение кортежа любого отношения в виде одной записи, в котором значения атрибута идут последовательно одно за другим, а за последним атрибутом кортежа в общем случае следует новый кортеж отношения.

Таким образом, на физическом носителе отношение R представлено в следующем виде:

$$\left[\dot{a}_{11},\dot{a}_{21},\ldots,\dot{a}_{n1}\right]_1\left[\dot{a}_{12},\dot{a}_{22},\ldots,\dot{a}_{n2}\right]_2\ \left[\dot{a}_{13},\dot{a}_{23},\ldots,\dot{a}_{n3}\right]_3\ldots$$

$$\ldots[\dot{a}_{1m},\dot{a}_{2m},\ldots,\dot{a}_{nm}]_m,$$ где $\dot{a}_{ij}$ — значение атрибута a_i. в j-м кортеже отношения R; $\left[\dot{a}_{1j},\dot{a}_{2j},\ldots,\dot{a}_{nj}\right]_j$ — j-й кортеж отношения R; n — количество атрибутов отношения R; $m = T(R)$ — количество кортежей отношения R.

В колоночных системах управления БД (СУБД) значения одного атрибута хранятся последовательно друг за другом [10], т.е. на физическом носителе отношение R примет следующий вид:

$$\left\langle\dot{a}_{11},\dot{a}_{12},\dot{a}_{13},\ldots,\dot{a}_{1m}\right\rangle_1\left\langle\dot{a}_{21},\dot{a}_{22},\dot{a}_{23},\ldots,\dot{a}_{2m}\right\rangle_2\ldots\left\langle\dot{a}_{n1},\dot{a}_{n2},\dot{a}_{n3},\ldots,\dot{a}_{nm}\right\rangle_n,$$

где $\dot{a}_{ij}$ — значение атрибута a_i в j-м кортеже отношения R;

$\left\langle\dot{a}_{i1},\dot{a}_{i2},\dot{a}_{i3},\ldots,\dot{a}_{im}\right\rangle_i$ — i-й столбец (атрибут) отношения R.

Каждая колонка, хранимая на диске, разделена на блоки определенного размера S_b. Блок состоит из заголовка, размер которого пренебрежительно мал по сравнению с размером блока и непосредственно данных. При одном запросе к диску осуществляется чтение нескольких блоков, количество которых определяется параметром.

Каждой записи в столбце относится ее позиция (номер строки). В большинстве современных колоночных БД [14] значения столбца упорядочиваются по их позициям.

На логическом уровне колоночные и строчные СУБД идентичны, т. е. способны обрабатывать одни и те же SQL-запросы. Но отличия в физической организации хранения данных существенно влияют на реализацию процессов, связанных с формированием плана выполнения запроса и его реализацией.

В строчных СУБД план запроса представляет собой дерево, у каждого узла которого имеется один родитель и один (или два в случае пересечения) дочерних узла. Реализация исполнителя планов базируется на следующих трех базовых парадигмах [15]: синхронный конвейер, итераторная модель, скобочный шаблон.

Более подробно изменения, вносимые в каждый из перечисленных элементов плана, рассмотрены в работах [11, 15].

Организация параллельной обработки данных. Основная форма параллельной обработки запросов в строчных и колоночных СУБД — фрагментный параллелизм. Подробно данный процесс рассмотрен в работах [2–4, 16]. В соответствие 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, которые реализуют те же операции, что и последовательный план (рис.1).

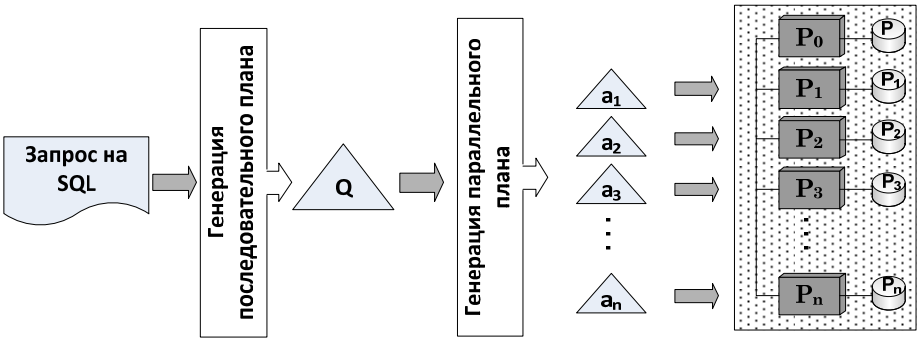

Рис. 1. Генерация параллельного плана запроса

Здесь n обозначает количество процессорных узлов. Это достигается путем вставки в соответствующие места дерева плана запроса составного оператора exchange, состоящего из четырех операторов: split, gather, merge и scatter. На завершающем этапе агенты рассылаются на определенные процессорные узлы, где интерпретируются исполнителем запросов. Результаты выполнения агентов объединяются корневым оператором exchange на нулевом процессорном модуле.

Рассмотрим процесс параллельной обработки запроса, где выполняется соединение таблиц R и S базы данных (рис. 2). $Q = R \rhd\lhd S$ — это логическая операция соединения (join) двух отношений (таблиц) R и S по некоторому общему атрибуту Y. В данном примере таблица R фрагментирована произвольным образом, а таблица S — по атрибуту соединения Y. На рис. 2 показано, что логический план выполнения соединения двух отношений тиражируется на n процессоров в параллельной системе баз данных (ПСБД) (на рисунке показаны два процессора). Далее происходит параллельная обработка на каждом процессоре соответствующих фрагментов таблиц R и S. Вследствие того, что таблица R не фрагментирована по атрибуту соединения, при последовательном чтении записей этой таблицы происходит их обработка в операторе exchange, осуществляющем разбор записи и ее межпроцессорный обмен. Таблица S фрагментирована по атрибуту соединения, и записи, читаемые из фрагментов этой таблицы, обрабатываются на каждом процессоре локально.

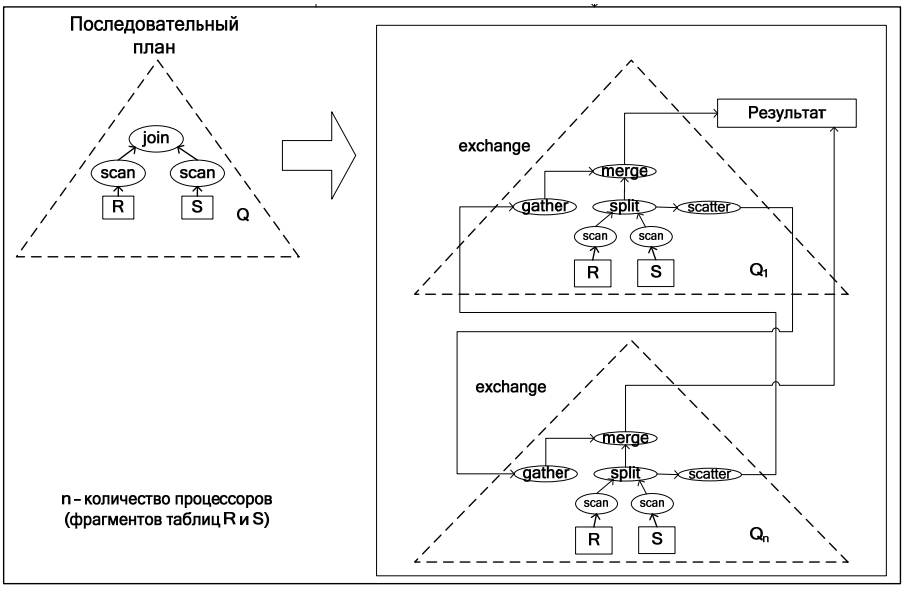

Рис. 2. Обработка запроса $Q = R \rhd\lhd S$ в параллельной системе баз данных

Наиболее распространенной системой классификации параллельных систем БД является система, предложенная Майклом Стоунбрейкером (Michael Stonebraker) [16]:

- SE (Shared-Everything) архитектура с разделяемыми памятью и дисками;
- SD (Shared-Disks) архитектура с разделяемыми дисками;
- SN (Shared-Nothing) архитектура без совместного использования ресурсов.

Обработка запроса к хранилищу данных в ПКБД. Процесс выполнения запроса к хранилищу данных, в частности к хранилищу со звездообразной схемой, может включать следующие шаги:

1) выделение множества кортежей в таблице фактов с использованием предикатов ограничений над одной или несколькими таблицами измерений;

2) выполнение некоторого агрегирования значений фактов, часто с группировкой по атрибутам таблицы измерений.

Таким образом, требуется выполнять соединения таблицы фактов и таблиц измерений для каждого предиката и каждой агрегатной группировки [17]. В качестве специфичного для колоночных БД плана запроса авторы работ [10, 15] предлагают метод, названный ими методом скрытых соединений, который можно использовать в системах БД с хранением данных по столбцам для соединений таблиц БД со звездообразной схемой по атрибутам *внешний-ключ/первичный-ключ*. Это соединение с отложенной материализацией, но в нем минимизируется число значений, которые требуется извлекать не в порядке следования позиций.

При использовании названного метода соединения выполняются в три этапа. Сначала каждый предикат применяется к соответствующей таблице измерений для извлечения списка ключей записей, удовлетворяющих данному предикату.

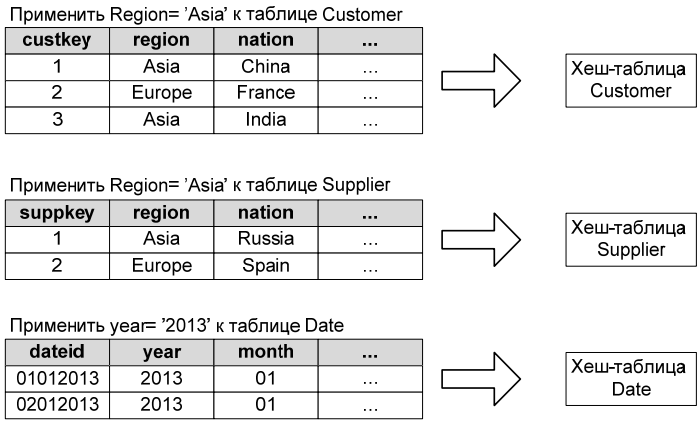

Рис. 3. Первый этап скрытого соединения

Эти ключи используются для построения хэш-таблицы, которую можно использовать для проверки того, удовлетворяет ли предикату некоторое значение ключа. Пример выполнения первого этапа показан на рис. 3.

На втором этапе хэш-таблицы используются для извлечения позиций тех записей из таблицы фактов, которые удовлетворяют соответствующему предикату. Для каждого значения столбца внешнего ключа таблицы фактов выполняется поиск в соответствующей хэш-таблице. Далее создается список всех позиций в этом столбце, значения которых удовлетворяют предикату. Затем списки позиций всех столбцов пересекаются и создается список позиций таблицы фактов, которые соответствуют записям, удовлетворяющим исходному условию поиска. Пример выполнения второго этапа показан на рис. 4.

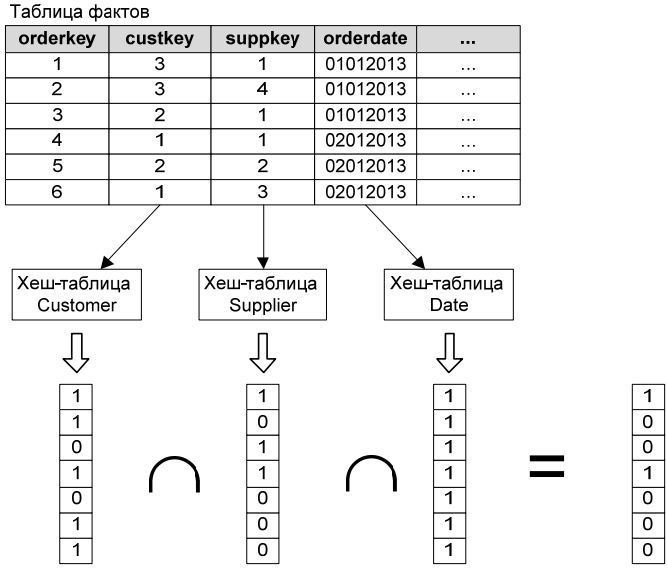

Рис. 4. Второй этап скрытого соединения

На третьем этапе с помощью списка позиций таблицы фактов осуществляется поиск в соответствующей таблице измерений. Если ключи таблицы измерений образуют отсортированный непрерывный список идентификаторов, начинающийся с единицы, значение внешнего ключа в действительности задает позицию нужного кортежа в таблице измерений. Это означает, что требуемые столбцы таблицы измерений могут быть извлечены напрямую с использованием этого списка значений внешнего ключа.

Пример выполнения третьего этапа показан на рис. 5. Для таблицы Date столбец ключа не является отсортированным непрерывным списком, начинающимся с единицы, поэтому для него требуется вы-

полнять полное соединение. Поскольку это соединение вида внешний-ключ/первичный-ключ и все предикаты уже применены, гарантируется, что в каждой таблице измерений для каждой позиции окончательного списка позиций таблицы фактов будет обнаружен один и только один результат. Это означает, что на этом третьем этапе при соединении с каждой таблицей измерений получается одно и то же число результатов, так что каждое соединение может выполняться по отдельности, и результаты могут материализоваться в более поздней точке плана выполнения запроса.

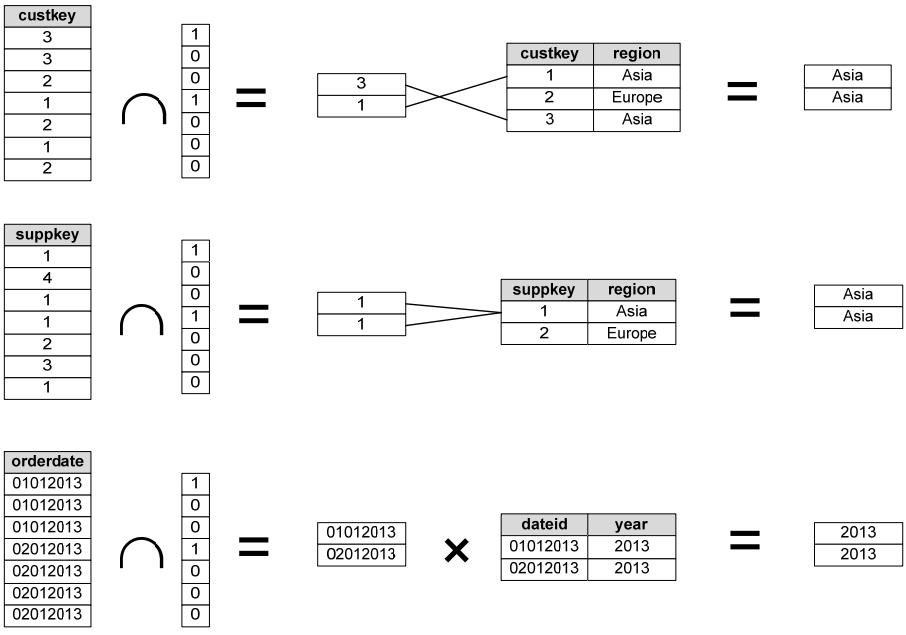

Рис. 5. Третий этап скрытого соединения

Преобразование Лапласа — Стилтьеса времени обработки запроса к хранилищу данных. Ниже поэтапно выводится преобразование Лапласа-Стилтьеса (ПЛС) времени выполнения запроса к хранилищу данных, которое справедливо для каждого узла параллельной колоночной системы баз данных. Здесь используется математический аппарат, изложенный в работах [2–6].

Все время обработки можно представить в виде суммы времени каждого из описанных ранее этапов.

Чтение ключевых атрибутов измерений. Введем следующие выражения для описания ПЛС времени K чтения таблиц измерений $(R_k, k=1, K)$ с условиями $F_{Rk} = f_0 \cap \bigcap_{i=1}^{|K_{F_{Rk}}|} f_i$ и пересылки ключевых атрибутов остальным узлам распределенной системы:

$$\Omega(P, z) = 1 - P(1 - z); \tag{1}$$

$$G(z) = z^{Vk/n} \tag{2}$$

— производящая функция (ПФ) числа позиций (записей) таблицы (отношения) *k*-го измерения, обрабатываемых на одном процессоре, где V_k — общее число записей в таблице *k*-го измерения; *n* — число процессоров в кластере (или в машине);

$$\chi_i(s, r, m) = \varphi_{D_i}(s)\varphi_M^{m\nu_i}(s)\varphi_P^r(s) \tag{3}$$

— ПЛС времени обработки в ресурсах, где $\varphi_{D_i}(s)$ — ПЛС времени чтения кортежа *i*-го столбца с диска для *k*-го измерения; $\varphi_M^{m\nu_i}(s)$ —ПЛС времени сохранения атрибута в оперативной памяти (ОП) и его чтения в кэш процессора; ν_i – размер атрибута; $m\nu_i$ — число операций чтения/записи в оперативную память, необходимых для проверки условия по соответствующему атрибуту*; $\varphi_P^r(s)$ – ПЛС времени обработки кортежа столбца в процессоре; *r* — число логических операций, необходимых для проверки условия по соответствующему атрибуту.

Функция $\psi(s,z)$ учитывает, что для *k*-й таблицы измерений читаются кортежи колонок по позициям, удовлетворяющим условиям поиска по предыдущим атрибутам (процесс поздней материализации, см. [12, 18]). Эта функция рекуррентно определяется следующим образом:

$$\begin{aligned}
\Psi(s, z) &= \Omega(1, \chi_1(s, r_1, m_1)\Psi_1(s, z)); \\
\Psi_1(s, z) &= \Omega(P_{f1}, \chi_2(s, r_2, m_2)\Psi_2(s, z)); , \\
\Psi_i(s, z) &= \Omega(P_{fi}, \chi_{i+1}(s, r_{i+1}, m_{i+1})\Psi_{(i+1)}(s, z)); \\
\Psi_b(s, z) &= \Omega(P_{fb}, \varphi_P^u(s)z)),
\end{aligned} \tag{4}$$

где $b = |K_{FRk}|$ — мощность подмножества атрибутов отношения, по которым происходит фильтрация кортежей по условию $\bigcap_{i=1}^{|K_{FRk}|} f_i$ для *k*-го измерения;

---

* Будем считать, что кэш процессора — это черная дыра, в которой сохраняются данные процессора, необходимые для вычислений. Из кэша в ОП перемещаются только результирующие материализованные записи, передаваемые по шине (см. далее $\varphi_N(s)$) процессору, где выполняется сборка.

$\varphi_P^u(s)$ учитывает, что для проверки условия f_0 для материализованных записей k-го измерения потребуется u логических операций процессора; P_{fi} — вероятность, что кортеж i-го атрибута удовлетворяет условию f_i.

Используя (1–4), получим ПЛС времени обработки k-го измерения на i-м узле:

$$R(s,\overline{z},z) = G(\Psi(s,\Omega(P_T,\chi_\pi(s,1,m)\overline{z}^{n-1}\cdot z))), \qquad (5)$$

где $\chi_\pi(s, 1, m)$ — ПЛС времени чтения кортежа ключевого атрибута с диска в кэш процессора и обработки в нем (см. (3)); 1 означает, что в процессоре проверяется только значение битовой маски в позиции, указанной в кортеже z — учитывает обработку в узле; $\overline{z}$ учитывает передачу данных в остальные $n-1$ узел; P_T — вероятность, что сформированный кортеж удовлетворяет условию f_0.

Тогда ПЛС времени K чтения таблиц измерений (R_k, $k = 1, K$) с условиями $F_{Rk} = f_0 \cap \bigcap_{i=1}^{|K_{FRk}|} f_i$ и пересылки ключевых атрибутов остальным узлам распределенной системы примет следующий вид:

$$D_i(s) = \prod_{k=1}^{K}\left[R_{ki}\left(s, \varphi_N^{w_k \nu_k}(s), \varphi_M^{w_k \nu_k}(s)\varphi_P^{hash}(s)\right)\times \right.$$
$$\left. \times \prod_{j=1,\ j\neq i}^{n} R_{kj}(0,1,\varphi_M^{w_k \nu_k}(s)\cdot\varphi_P^{hash}(s))\right], \qquad (6)$$

где K — общее количество таблиц измерений, участвующих в запросе; $R_{ki}(s)$, $R_{kj}(s)$ — ПЛС времени обработки k-го измерения на i-м (j-м) узле; $\varphi_N^{w_k \nu_k}(s)$, $\varphi_M^{w_k \nu_k}(s)$ — учитывает перемещение ключевых атрибутов k-го измерения, удовлетворяющих условию поиска F; ν_k — размер сформированного кортежа; $w_k \nu_k$ — количество операций чтения/записи, необходимое для перемещения сформированных записей ключевых атрибутов измерений; $\varphi_P^{hash}(s)$ учитывает процессорное время на создание хеш-таблиц.

Извлечение битовой маски таблицы фактов. Введем следующие выражения для описания ПЛС времени чтения данных таблицы фактов, получения битовой маски и передачи ее всем узлам распределенной сети:

$$G_f(z) = z^{V/n} \qquad (7)$$

— ПФ числа позиций (записей) таблицы фактов, обрабатываемых на одном процессоре, где V — общее число записей в таблице фактов.

Функция $\Psi(s,z)$ учитывает, что на основе полученных ключевых атрибутов таблиц измерений и условий фильтрации происходит чтение атрибутов таблицы фактов, и определяется по формуле

$$\Psi(s,z) = \Psi_R(s, \Psi_F(s,z)), \qquad (8)$$

где функции ψ_R и ψ_F рекуррентно определяются следующим образом:

$$\Psi_R(s,z) = \Omega(P_1, \chi_1(s, r_1, m_1)\Psi_1(s,z));$$

$$\Psi_1(s,z) = \Omega(P_2, \chi_2(s, r_2, m_2)\Psi_2(s,z)), \qquad (9)$$

...

$$\Psi_K(s,z) = \Omega(P_K, \chi_K(s, r_K, m_K)\, z).$$

Здесь K — количество таблиц-измерений, участвующих в запросе; P_k — вероятность, что кортеж атрибута удовлетворяет условию k-го измерения в таблице фактов ($\prod_{i=1}^{b} P_{fi} P_T$).

В формуле (10) и далее P_{fi} — вероятность, что кортеж считываемой колонки таблицы фактов удовлетворяет соответствующему предикату:

$$\Psi_F(s,z) = \Omega(P_{f1}, \chi_1(s, r_1, m_1)\Psi_1(s,z));$$

$$\Psi_1(s,z) = \Omega(P_{f2}, \chi_2(s, r_2, m_2)\Psi_2(s,z)); \qquad (10)$$

...

$$\Psi_i(s,z) = \Omega(P_{f_i}, \chi_{i+1}(s, r_{i+1}, m_{i+1})\Psi_{(i+1)}(s,z));$$

...

$$\Psi_a(s,z) = \Omega(P_{f_a}, \varphi_P^u(s)\, z)),$$

где a — количество атрибутов таблицы фактов, по которым происходит фильтрация кортежей (ограничение на факты); $\varphi_P^u(s)$ учитывает, что для проверки условия f_0 для материализованных записей таблицы фактов потребуется u логических операций процессора; P_{fa} — вероятность, что сформированный кортеж удовлетворяет условию f_0.

Используя (7–10), получим ПЛС времени обработки таблицы фактов на i-м узле:

$$J(s, \overline{z}, z) = G_f(\Psi(s, \Omega(P_T, \chi_\pi(s, 1, m)\overline{z}^{n-1} z))), \qquad (11)$$

где $\overline{z}$ — учитывает передачу полученной маски в остальные n–1 узел для чтения необходимых атрибутов таблиц измерений (шаг 3 скрытого соединения).

Тогда ПЛС времени чтения данных таблицы фактов, получения битовой маски и передачи ее всем узлам распределенной сети примет следующий вид:

$$M_i(s) = J_i(s, \varphi_N^{w_F \nu_F}(s), \varphi_M^{w_F \nu_F}(s)) \times \prod_{j=1, j\neq i}^{n} J_k(0,1,\varphi_M^{w_F \nu_F}(s)), \qquad (12)$$

где $\varphi_N^{\nu_F}(s)$, $\varphi_M^{\nu_F}(s)$ учитывает перемещение ключевых атрибутов, удовлетворяющих условию поиска; ν_F — размер передаваемой битовой маски.

Чтение значений атрибутов измерений. Для получения ПЛС времени чтения дополнительных атрибутов измерений, если такие присутствуют в запросе, введем следующие выражения:

$$G(z)=z^{Vk/n}; \qquad (13)$$

$$H(s,\overline{z},z) = G(\Omega(P, \prod_{j=1}^{A_k} \chi_j(s,1,m))\,\overline{z}^{n-1} z), \qquad (14)$$

где P — вероятность, что итоговый кортеж удовлетворяет условиям поиска $\left(\prod_{k=1}^{K} P_k P_{fact}\right)$; A_k — количество атрибутов k-й таблицы измерений, необходимых для предоставления результата выполнения запроса (то, что указывается в SELECT).

Тогда ПЛС времени чтения дополнительных атрибутов измерений, если такие присутствуют в запросе, примет следующий вид:

$$U_i(s) = \prod_{k=1}^{K} \left[H_{ki}(s, \varphi_N^{w_k \nu_k}(s), \varphi_M^{w_k \nu_k}(s)) \times \right.$$

$$\left. \times \prod_{j=1, j\neq i}^{n} H_{kj}(0,1,\varphi_P^{mat}(s)\varphi_M^{w\nu}(s)\varphi_N^{w\nu}(s)) \right], \qquad (15)$$

где $\varphi_P^{mat}(s)$ учитывает время, необходимое на воссоздание кортежа результата; $\varphi_N^{w\nu}(s)$, $\varphi_M^{w\nu}(s)$ учитывает перемещение результатов запроса.

ПЛС времени обработки кортежей в ресурсах. Формулы для $\varphi_{D_i}(s)$ (i — номер колонки, т. е. атрибута), $\varphi_M(s)$, $\varphi_N(s)$, $\varphi_P(s)$ в за-

висимости от архитектуры параллельных колоночных систем баз данных (ПКСБД) и режима работы представлены в таблице [14].

Таблица

<table>
<tr><th colspan="2"></th><th>$\varphi_{D_i}(s)$</th><th>$\varphi_M(s)$</th><th>$\varphi_N(s)$</th><th>$\varphi_P(s)$</th></tr>
<tr><td rowspan="3">Online</td><td>SE</td><td rowspan="3">$(1-\frac{1}{L_i})+\frac{1}{L_i}((1-p_D)+$
$+p_D\frac{\mu_{DB}-\lambda_{DB}}{\mu_{DB}-\lambda_{DB}+s})$</td><td colspan="2">$\frac{\mu_M-\lambda_M}{\mu_M-\lambda_M+s}$
(обмен между процессорами осуществляется через ОП)</td><td rowspan="3">$\frac{\mu_P-\lambda_P}{\mu_P-\lambda_P+s}$</td></tr>
<tr><td>SD</td><td rowspan="2">$\frac{\mu_M-\lambda_M}{\mu_M-\lambda_M+s}$</td><td rowspan="2">$\frac{\mu_N-\lambda_N}{\mu_N-\lambda_N+s}$</td></tr>
<tr><td>SN</td></tr>
<tr><td colspan="2">Offline</td><td>$(1-\frac{1}{L_i})+\frac{1}{L_i}((1-p_D)+p_D\eta_{DB}(s))$
$\eta_{DB}(s)=\frac{\Theta^n(\Omega(\frac{1}{N_D},\phi_{DB}(s)))}{\Omega(\frac{1}{N_D},\phi_{DB}(s))}\phi_{DB}(s)$
$\phi_{DB}(s)=\frac{\mu_{DB}}{\mu_{DB}+s}$</td><td>$\frac{\mu_M}{\mu_M+s}$</td><td>$\frac{\mu_N}{\mu_N+s}$</td><td>$\frac{\mu_P}{\mu_P+s}$</td></tr>
</table>

При выводе учитывались следующие особенности выполнения запроса в колоночной СУБД [11]:

- каждая колонка хранится на диске в своих блоках, где отдельная колонка представляет собой таблицу с кортежем (значение атрибута, позиция);
- последовательная и параллельная обработка запросов с поздней материализацией кортежей;
- наличие компрессии данных (метод RLE [19]);
- получение времени работы обслуживающих устройств на основе измеримых с помощью синтетических тестов показателей.

При этом рассматривались два режима работы [14]:

1. Пакетный режим (offline, система рассматривается как замкнутая).

При данном режиме работы в колоночной системе баз данных обрабатываются пакеты запросов. В каждом пакете SQL-запросы выполняются последовательно (предполагается, что они связаны по данным: выходные данные одного запроса являются входными данными другого). Но запросы разных пакетов (по одному из каждого пакета) могут обрабатываться параллельно. Предполагается, что «узкое место» в данном режиме — дисковая подсистема.

2. Режим «запрос-ответ» (online, система рассматривается как разомкнутая). При данном режиме работы предполагается, что *i*-я рабочая станция обращается к *j*-му запросу с некоторой интенсивностью. При условии, что эти входные потоки заявок являются пуассоновскими, время обслуживания в ресурсах распределено по экспоненциальному закону, а переход от ресурса к ресурсу выполняется по вероятности, модель обработки запросов можно представить в виде сети массового обслуживания. В такой сети обработку в узлах ресурсов можно представить в виде совокупности независимых систем массового обслуживания M/M/1 (это доказывается в теории массового обслуживания в виде теоремы разложения Джексона).

Итоговое ПЛС времени обработки запроса к ПКХД. Итоговое ПЛС времени обработки запроса к ПКХД в i-м узле приведено ниже:

$$\varphi_i(s) = D_i(s)M_i(s)U_i(s)\varphi_P^{agr}(s)\varphi_M^{agr}(s), \qquad (16)$$

где $D(s)$, $M(s)$, $U(s)$ — ПЛС времени обработки соответственно первого, второго и третьего этапов скрытого соединения (см. (6), (12), (15)); $\varphi_P^{agr}(s)\varphi_M^{agr}(s)$ — учитывают время на агрегацию данных, если это указано в запросе.

Дифференцируя выражение (16) как сложную функцию по s в нуле, можно получить моменты случайного времени ξ обработки запроса к ПКХД:

$M_\xi = -\varphi'(0)$, $M_{\xi^2} = \varphi''(0)$, $\sigma_\xi^2 = M_{\xi^2} - M_\xi^2$.

Для получения значений моментов можно использовать методы численного дифференцирования, описанные в [20]:

$$\varphi'(x_0) \approx \frac{(\varphi_{1/2}^1 - \frac{1}{2}\varphi_1^2)}{h}; \qquad (18)$$

$$\varphi''(x_0) \approx \frac{(\varphi_0^2 - \frac{1}{12}\varphi_0^4)}{h^2}, \qquad (19)$$

где $$\varphi_i^m = \sum_{j=0}^{m} (-1)^j C_m^j \varphi_{i+m/2-j} \qquad (20)$$

— разность m-го порядка (при четном *m i* — целое, при нечетном — полуцелое);

$$\varphi_i = \varphi(x_i) = \varphi(x_0 + ih) \qquad (21)$$

– соответствующие значения функции; *h* — шаг таблицы разностей.

Сравнение производительности соединения методом NLJ и скрытого соединения в ПКХД. Ниже приведено сравнение времени выполнения запроса к хранилищу данных для скрытого соединения и соединения методом NLJ [14]. Характеристики ресурсов (интенсивности обработки) были получены с помощью программы синтетических тестов AIDA64 [21]. Расчеты были выполнены при следующих значениях характеристик ресурсов.

1. Процессор Intel Core i7-920 2.79GHz. Для выбранного процессора измеренное значение числа процессорных циклов, выполняемых в секунду, $\mu_P = 2{,}79 \cdot 10^9$(1/с).

2. Внешняя память N_D=250, диск 3,5" Seagate Cheetah 15K.6 ST3146356FC; размер блока чередования (stripe size) (БЧ) $Q_{БЧ}$ = 64 Кб; среднее время поиска и чтения блока чередования с диска $t_{БЧ} = t_{подвода} + t_{вращения}/2 + Q_{БЧ}/v_{чтения} = 4 + 4/2 + 64/200 = 6.3$ мс. Поэтому интенсивность чтения блоков с диска μ_{DB}= 1000/6,3= 160 (1/с), p_D=0,9.

3. Оперативная память DDR3-1600 PC3 — 12 800. Интенсивность чтения одного байта информации из ОП μ_M= 9586·1024·1024 (1/с).

В качестве примера был выбран аналитический запрос *Q*3 теста TPC-H [22]. Схема базы данных тестовой среды приведена на рис. 6. Коэффициент *sf* теста определяет объем обрабатываемых данных:

```
select l_orderkey,
 sum(l_extendedprice*(1-l_discount)) as revenue,
 o_orderdate,
 o_shippriority
from customer,
 orders,
 lineitem
where c_mktsegment = '[SEGMENT]'
 and c_custkey = o_custkey
 and l_orderkey = o_orderkey
 and o_orderdate < date '[DATE]'
 and l_shipdate > date '[DATE]'
```

group by l_orderkey,
o_orderdate,
o_shippriority
order by revenue desc,
o_orderdate

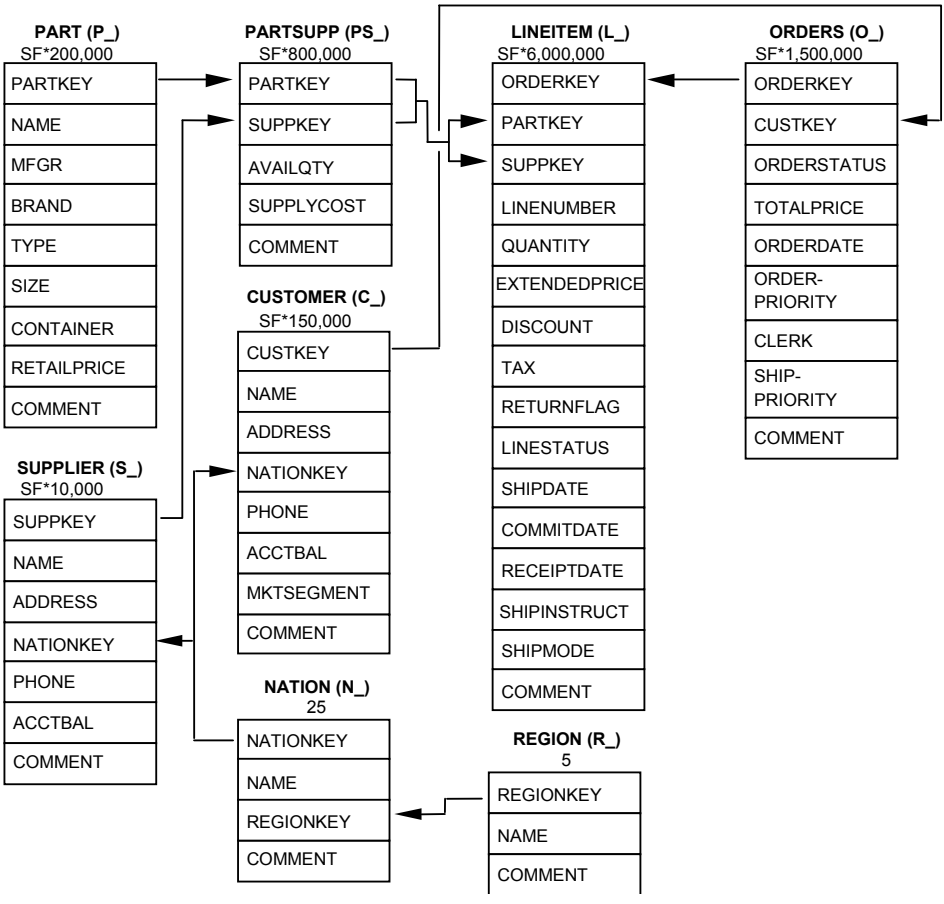

Рис. 6. Схема базы данных теста TPC-H

График зависимости времени выполнения запроса от количества узлов для скрытого соединения и соединения методом NLJ для архитектуры SE представлен на рис. 7. Из графика видно, что метод скрытого соединения превосходит по скорости метод NLJ, причем соотношение времени выполнения сохраняется при изменении количества узлов.

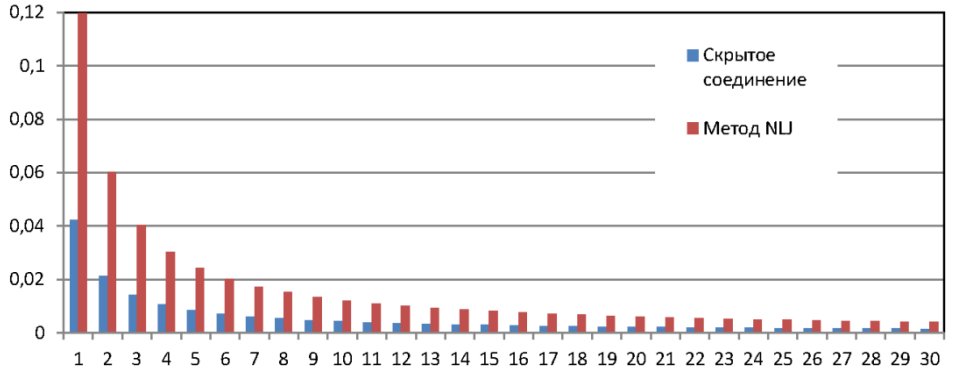

Рис 7. Зависимость времени выполнения запроса для различных методов соединения при архитектуре SE (*sf* = 0,01)

На рис. 8 приведена зависимость времени выполнения запроса для различных методов соединения и архитектур от коэффициента *sf* теста TPC-H (*sf*). При увеличении объема обрабатываемых данных время обработки при использовании метода NLJ увеличивается быстрее, чем при методе скрытого соединения.

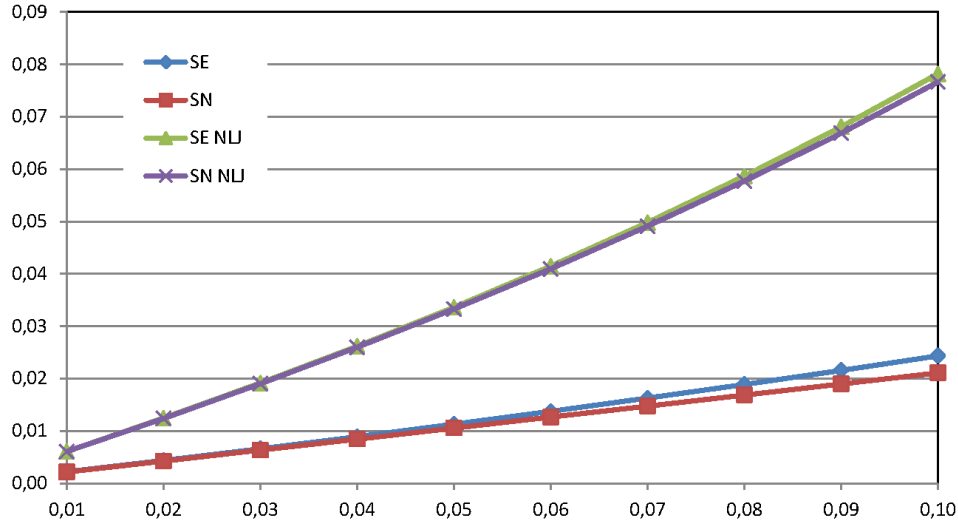

Рис 8. Зависимость времени выполнения запроса для различных методов и архитектур от коэффициента *sf* теста TPC-H (*n*=20)

Таким образом, авторами

- проанализирован способ выполнения запросов в параллельном колоночном хранилище данных, рассмотрен специфичный для колоночных систем процесс «скрытого соединения» (invisible join);
- получено преобразование Лапласа — Стилтьеса времени выполнения запроса со скрытой материализацией, рассмотрены

варианты этого преобразования для различных архитектур параллельных систем баз данных;

- проведено сравнение времени выполнения запроса к хранилищу данных для скрытого соединения и соединения методом NLJ, показано превосходство метода скрытого соединения.

Из изложенного выше можно сделать вывод, что

- полученное соотношение увеличения скорости выполнения запроса (в 2.75 раза) соответствует экспериментально полученному в работе [15] значению;
- авторы предполагают продолжить исследования и получить модель адаптации полученной ПЛС к конкретной реализации ПКХД.

ЛИТЕРАТУРА

[1] Арсентьев А. Хранилища данных становятся инфраструктурным компонентом № 1. *CNews аналитика*, 2010.
URL: http://retail.cnews.ru/reviews/free/BI2010/articles/articles6.shtml (дата обращения 27.06.2011).

[2] Григорьев Ю.А., Плутенко А.Д. *Теоретические основы анализа процессов доступа к распределенным базам данных*. Новосибирск, Наука, 2002, 222 с.

[3] Григорьев Ю.А., Плужников В.Л. *Оценка времени выполнения запросов и выбор архитектуры параллельной системы баз данных*. Москва, МГТУ им. Н.Э. Баумана, 2009.

[4] Григорьев Ю.А., Плужников В.Л. Модель обработки запросов в параллельной системе баз данных. *Вестник МГТУ им. Н.Э. Баумана*, 2010, № 4, с. 78–90.

[5] Григорьев Ю.А., Плужников В.Л. Оценка времени соединения таблиц в параллельной системе баз данных. *Информатика и системы управления*, 2011, № 1, с. 3–16.

[6] Григорьев Ю.А., Плужников В.Л. Анализ времени обработки запросов к хранилищу данных в параллельной системе баз данных. *Информатика и системы управления*, 2011, № 2, с. 94–106.

[7] Stonebraker M., Çetintemel U. One Size Fits All: URL: http://citforum.ru/database/articles/one_size_fits_all/. 27.06.2011.

[8] Stonebraker M., Bear C., Çetintemel U., Cherniack M., Ge T., Hachem N., Harizopoulos S., Lifter J., Rogers J., Zdonik S. One Size Fits All? Part 2: Benchmarking Results. 3rd Biennial Conference on Innovative Data Systems Research (CIDR), January 7–10, 2007, Asilomar, California, USA. URL: http://citforum.ru/database/articles/one_size_fits_all_2/ (дата обращения 27.06.2011).

[9] Stonebraker M. My Top 10 Assertions About Data Warehouses. URL: http://citforum.ru/gazeta/166/]. 27.06.2011.

[10] Stonebraker M., Abadi D.J., Batkin A., Chen X., Cherniack M., Ferreira M., Lau E., Lin A., Madden S.R., O'Neil E.J., O'Neil P.E., Rasin A., Tran N., S.B. Zdonik: C-Store: A Column-Oriented DBMS URL: http://www.cs.yale.edu/homes/dna/pubs/displaypubs.cgi/ (дата обращения 22.10.2011)

[11] Григорьев Ю.А., Ермаков Е.Ю. Модель обработки запросов в параллельной колоночной системе баз данных. *Информатика и системы управления*, 2012, № 1, с. 3–15.
[12] Григорьев Ю.А., Ермаков Е.Ю. Модель обработки запроса к одной таблице в параллельной колоночной системе баз данных и анализ ее адекватности. *Информатика и системы управления*, 2012, № 2. с. 170–179.
[13] Григорьев Ю.А., Ермаков Е.Ю. Сравнение процессов обработки запроса к одной таблице в параллельной строчной и колоночной системе баз данных. *Вестник МГТУ им. Н.Э. Баумана*. Спец. выпуск; № 5, 2012, с. 31–45.
[14] Григорьев Ю.А., Ермаков Е.Ю. Оценка времени соединения двух таблиц в параллельной колоночной системе баз данных // Вестник МГТУ им. Н.Э. Баумана — 2012 - № 4. — С. 80-100.
[15] Daniel J. Abadi Query Execution in Column-Oriented Database Systems. [Электронный ресурс]. http://www.cs.yale.edu/homes/dna/papers/abadiphd.pdf (дата обращения 25.12.2011).
[16] Соколинский Л. Б., Цымблер М. Л. Лекции по курсу «Параллельные системы баз данных»: [Электронный ресурс]. http://pdbs.susu.ru/CourseManual.html (дата обращения 22.10.2011).
[17] Кузнецов С. СУБД с хранением данных по столбцами и по строкам: насколько они отличаются в действительности? [Электронный ресурс]. http://citforum.ru/database/articles/column_vs_row_store/ (Дата обращения 24.04.2012).
[18] Daniel J. Abadi, Daniel S. Myers, David J. DeWitt, and Samuel R. Madden. Materialization Strategies in a Column-Oriented DBMS *In Proceedings of ICDE,* 2007. [Электронный ресурс]. http://db.lcs.mit.edu/projects/cstore/abadiicde2007.pdf (дата обращения 25.12.2011).
[19] Daniel J. Abadi, Samuel R. Madden and Miguel C. Ferreira. Integrating Compression and Execution in Column-Oriented Database Systems *In Proceedings of ICDE, 2006.* [Электронный ресурс]. http://db.lcs.mit.edu/projects/cstore/abadisigmod06.pdf (дата обращения 25.12.2011).
[20] Бахвалов Н.С. Численные методы (анализ, алгебра, обыкновенные дифференциальные уравнения). - М.: Наука, 1973, 631 с.
[21] AIDA64 Extreme Edition. [Электронный ресурс] http://www.aida64.com/product/aida64-extreme-edition/overview (дата обращения 08.04.2012).
[22]Спецификация теста TPC-H. [Электронный ресурс] http://www.tpc.org/tpch/ (дата обращения 24.04.2013).

Статья поступила в редакцию 24.06.2013

Ссылку на эту статью просим оформлять следующим образом:

Григорьев Ю.А., Ермаков Е.Ю. Анализ времени выполнения запроса в параллельном колоночном хранилище данных. *Инженерный журнал: наука и инновации*, 2013, вып. 11. URL: http://engjournal.ru/catalog/it/hidden/1069.html

Григорьев Юрий Александрович — д-р техн. наук, проф. кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана. e-mail: grigorev@bmstu.ru

Ермаков Евгений Юрьевич — аспирант кафедры «Системы обработки информации и управления» МГТУ им. Н.Э. Баумана. e-mail: JK.Ermakov@gmail.com