УДК 004.657

© 2015 г. **Ю.А. Григорьев,** д-р техн. наук
(Московский государственный технический университет им. Н.Э. Баумана),
А.Д. Плутенко, д-р техн. наук
(Амурский государственный университет)

МЕТОД ОЦЕНКИ ВРЕМЕНИ ТЕТА-СОЕДИНЕНИЯ ТАБЛИЦ БАЗЫ ДАННЫХ ПО ТЕХНОЛОГИИ MAPREDUCE

Исследованы две стратегии копирования (дублирования) кортежей при выполнении многотабличного тета-соединения по технологии MapReduce: с использованием кривых Гильберта и интервальная стратегия. Определены варианты их преимущественного использования. Получены формулы для оценки времени выполнения тета-соединения, учитывающие процессорную, дисковую и сетевую составляющие. Рассмотрен практический пример, выявлен характер изменения процессорного времени в ходе соединения фрагментов таблиц в узле.
Ключевые слова: оценка времени, тета-соединение, технология MapReduce, кривая Гильберта, интервальная стратегия.

Введение

Многотабличное тета-соединение (Multi-way Theta-join) означает, что в соединении участвуют более двух таблиц базы данных и условие соединения может быть произвольным. Это мощный аналитический инструмент для анализа сложных корреляций данных. Операторы, участвующие в запросе многотабличного соединения – не только просто эквивалентное соединение таблиц. Условие тета-соединения может быть определено как бинарная функция

$\theta \in \{<, \leq, =, \geq, >, <>\}$.

В качестве примера рассмотрим следующий сценарий приложения [1]:

«*Допустим, у нас есть множество городов* $\{c_1, c_2, ..., c_K\}$ *и все данные о рейсах* FI_{ij} *между любыми двумя городами* c_i *и* c_j. *Задано некоторое подмножество городов* $< c_s, ..., c_t >$ *и время пребывания, которое должно находиться в интервале* $L_i = [l_1, l_2]$,*в каждом городе* c_i. *Необходимо определить все возможные туристические планы*».

Это практический запрос, который мог бы помочь путешественникам планировать свои поездки. Мы предполагаем, что FI – таблица, содержащая номер города отправления (id1), номер города назначения (id2), номер рейса, время отправления (dt), время прибытия (at), нижнюю границу времени пребывания в городе назначения (l1), верхнюю границу времени пребывания в городе назначения (l2). Приведенную выше задачу можно решить с помощью многотабличной опе-

рации тета-соединения последовательности кортежей $FI_{s,s+1}$,..., $FI_{t-1,t}$, указав, что время отправления рейса в следующий город должно находиться в интервале пребывания туристов в текущем город, т.е. θ-функции можно представить в виде следующих неравенств:

$FI_{s,s+1}.at+FI_{s,s+1}.l1<FI_{s+1,s+2}.dt<FI_{s,s+1}.at + FI_{s,s+1}.l2$.

Соответствующий оператор SELECT можно представить в следующем виде:

SELECT $s_1.id1, s_2.id1, \ldots, s_n.id1, s_n.id2$
FROM $FI\ s_1, FI\ s_2, \ldots, FI\ s_n$
WHERE $s_1.id1\ IN\ (c_1,..., c_{n+1})$ and $s_1.id2\ IN\ (c_1,..., c_{n+1})$ and $s_1.id2 <> s_1.id1$ and
$s_2.id1=s_1.id2$ and $s_2.id2\ IN\ (c_1,..., c_{n+1})$ and $s_2.id2 <> s_1.id1$ and $s_2.id2 <> s_2.id1$ and
…
$s_n.id1=s_{n-1}.id2$ and $s_n.id2\ IN\ (c_1,..., c_{n+1})$ and $s_n.id2 <> s_1.id1$ and
$s_n.id2 <> s_2.id1$ and … $s_n.id2 <> s_{n-1}.id1$and $s_n.id2 <> s_n.id1$ and
---
$s_1.at+ s_1.l1< s_2.dt$ and $s_2.dt <s_1.at+ s_1.l2$ and
$s_2.at+ s_2.l1< s_3.dt$ and $s_3.dt <s_2.at+ s_2.l2$ and
…
$s_{n-1}.at+ s_{n-1}.l1< s_n.dt$ and $s_n.dt <s_{n-1}.at+ s_{n-1}.l2$;

Здесь таблица FI соединяется сама с собой nраз, где (n + 1) – число городов в круизе. Первые строки в предикате WHERE (до условной штриховой линии) моделируют перестановку номеров городов $(c_1,...,c_{n+1})$, число таких перестановок не превышает (n+1)!. Но в рамках одной перестановки может быть несколько вариантов, связанных с разными рейсами. Остальные строки описывают рассмотренные выше θ-функции: время отправления рейса в следующий город должно находиться в интервале пребывания туристов в текущем городе. Этот пример показывает, что в тета-соединени может участвовать много таблиц (возможно в виде псевдонимов s_i одной и той же таблицы FI).

В ранних работах [2 – 4] были выявлены проблемы оценки стоимости выполнения таких запросов и представлены стратегии их решения. Однако эти решения не масштабируются для обработки многотабличного тета-соединения поверх данных огромных объемов. К примеру, в Facebook [5] и Google [6] основной объем данных имеет сотни терабайт или даже петабайт. В таких случаях решения на основе традиционных баз данных, распределенных или параллельных, неосуществимы из-за неудовлетворительной масштабируемости и отказоустойчивости.

В то же время модель программирования MapReduce гарантирует существенно большую масштабируемость и сильные свойства толерантности. Она стала самой популярной парадигмой обработки больших объемов данных в вычислительной среде с неразделяемыми ресурсами. В настоящее время сообщество исследователей баз данных сосредоточено главным образом на решении двух вопросов [1]. Во-первых, это переход от некоторых операторов реляционной алгебры (например, соединения) к параллельным вычислениям на основе <ключ, значение>. Во-вторых, это тюнинг (переработка) функций таким образом, чтобы задания MapReduce выполнялись более эффективно с точки зрения уменьшения затрат времени и потребления вычислительных ресурсов. При решении первой задачи применительно к тета-соединению необходимо учитывать, что записи таб-

лиц уже фрагментированы по узлам системы и необходимо выполнить их частичное дублирование по некоторым узлам некоторым эффективным способом. В этих узлах параллельно выполняется операция тета-соединения записей, поступивших в узлы. Для решения второй задачи предстоит разработать стоимостную модель реализации тета-соединения в узле.

Стратегии дублирования записей таблиц

При рассмотрении стратегий предполагается, что тета-соединение реализуется в рамках одного задания Job MapReduce (MR). Варианты, включающие несколько заданий, как правило, неэффективны [1].

Введем следующие обозначения:

N – число параллельно выполняющихся задач Reduce; будем полагать, что оно равно числу узлов в системе MR;

R_i – множество номеров кортежей i-й таблицы, участвующей в тета-соединении, i = 1,…,n;

номер кортежа можно считать его ключом;

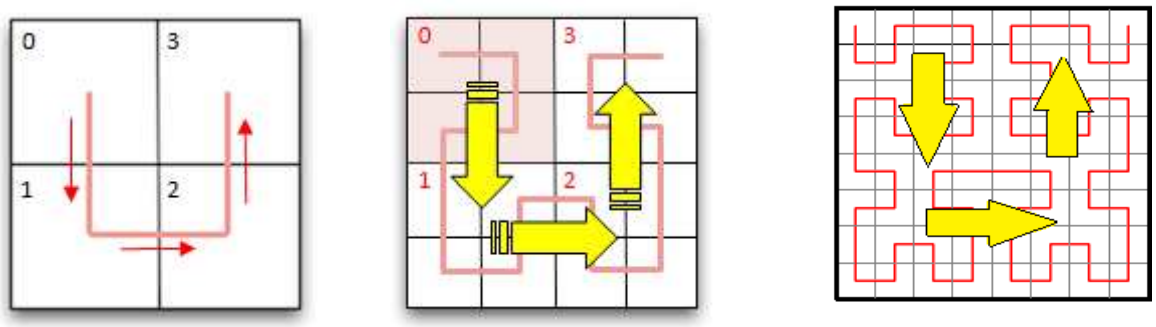
$|R_i|$ – число кортежей в i-й таблице;

$S = R_1 \times \ldots \times R_n$ – n-мерный куб, элементом которого является множество из «n»номеров кортежей (по одному из каждой таблицы);

$C = (c_1, \ldots, c_N)$ – разбиение куба S на подкубы: $c_i \cap c_j = \varnothing$, $\cup c_i = S$; кортежи измерений подкуба c_i обрабатываются (соединяются) на i-м узле.

Требуется найти такое разбиение C, чтобы общее число дублирований (перемещений) кортежей на этапе распространения результатов выполнения фазы Map(shuffle) было минимальным. Обозначим этот критерий как Ψ(C).

В работе [1] предложено для получения лучшего разбиения C_H использовать *кривые Гильберта*[7,8], которые пробегают все ячейки n-мерного куба S по одному разу. На рис. 1 показаны кривые Гильберта для двухмерного куба S.



а) б) в)

Рис. 1. Примеры кривой Гильберта.

Для варианта 2×2 (рис. 1а) кривая начинается в 0-й ячейке и заканчивается в 3-й ячейке. Как видно из рис. 1б (вариант 4×4), кривая проходит через квадранты 0÷3. Причем форма кривой в каждом квадранте повторяет форму кривой для размерности 2×2 (в каждом квадранте она переворачивается). Аналогично кривая для случая 8×8 (рис. 1в) представляет собой набор кривых для варианта 4×4 (см. рис. 1б). Сформулируем два свойства кривых Гильберта.

Свойство 1. Кривая для большей размерности представляет набор кривых меньшей размерности (по принципу вложенных матрешек).

Свойство 2. Кривая Гильберта покидает куб меньшей размерности, обойдя все ячейки этого куба (см. рис. 1 б, в).

Эти свойства сохраняются для исходного куба S произвольной размерности, если число кортежей $|R_i|$ кратно некоторой степени 2, т.е. 2^m. Действительно, в этом случае $S \subseteq 2^M \times \ldots \times 2^M$ и при обходе ячеек «большого» куба размерности $\left(2^M\right)^n$ кривая Гильберта обойдет и все кубы меньшей размерности $\left(2^m\right)^n$, которые составляют S.

Рассмотренную выше стратегию разбиения на основе кривой Гильберта также будем обозначать как C_H. В [1] доказывается теорема, в соответствии с которой кривая Гильберта оптимально разбивает пространство S, т.е. число дублирований кортежей является минимальным. Доказательство выполняется с предпосылкой, что минимум достигается в случае равенства числа дублирований кортежей разных таблиц R_i. В нашей статье мы докажем более сильное утверждение.

Разобьем номера кортежей каждой таблицы на последовательные интервалы $\{NR_{ij}\}$, где i – номер таблицы; j – номер интервала.

Разбиение C_I будем называть *интервальным*, если любой подкуб c_i можно представить в следующем виде: $c_i = NR_{1i_1} \times \ldots \times NR_{ni_n}$, и $\forall m \exists i, j\left(\left|NR_{ij}\right| \neq 2^m\right)$ (рис. 2).

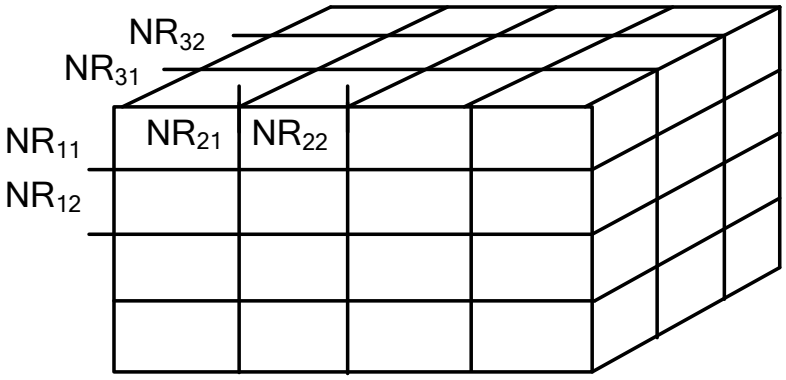


Рис. 2. Интервальное разбиение куба.

В работе [9] предлагается использовать оптимальное интервальное разбиение пространства S для реализации тета-соединения. Так какая из стратегий лучше – C_H или C_I? Докажем сначала следующую теорему.

Теорема 1. Разбиение куба Sc помощью кривой Гильберта лучше любого интервального разбиения по критерию общего числа дублирования кортежей, т.е. $\Psi(C_H) \leq \Psi(C_I)$ для $\forall C_I$. Причем равенство достигается только если $|R_i|/N_i = |R_j|/N_j$ для $\forall i, j$; N_j – это число интервалов, на которое разбивается j-е измерение (записи j-й таблицы) при использовании интервальной стратегии.

Доказательство. Сначала определим общее число дублирований кортежей по всем узлам сети, если для разбиения исходного куба используется кривая Гильберта. Имеет место равенство:

$$\frac{\prod_{j=1}^{n}|R_j|}{N} = \left(2^m\right)^n, \qquad (1)$$

где N – число областей, на которые разбивается исходный куб S с помощью кривой Гильберта; 2^m – число записей какого-либо измерения (таблицы) в одной области (подкубе). Каждый подкуб обрабатывается в отдельном узле, поэтому N – это число узлов в системе.

Равенство (1) следует из того, что левая и правая его части определяют число элементов в каком-либо подкубе $c_к$. Это следует из свойств 1 и 2 кривой Гильберта.

Число интервалов, на которое разбивается каждое j-е измерение, определяется следующим выражением:

$$q_j = \frac{|R_j|}{2^m}, j = 1 \ldots n. \qquad (2)$$

Будем полагать, что q_j – целое число. Если это не так, то к таблице R_j можно добавить требуемое число пустых записей, которые не участвуют в операции соединения и не пересылаются на другие узлы.

Общее число дублирований кортежей по всем узлам сети равно

$$\Psi(C_H) = \sum_{i=1}^{n}|R_i|\prod_{j\neq i}^{n} q_j. \qquad (3)$$

Действительно, произведение в (3) определяет число областей (узлов) в сечении исходного куба, связанном с одной записью какой-либо таблицы R_i. Используя выражения (1), (2) и (3), получим

$$\Psi(C_H) = n \cdot N \cdot \sqrt[n]{\frac{\prod_{i=1}^{n}|R_i|}{N}}. \qquad (4)$$

Теперь определим общее число дублирований кортежей по всем узлам сети, если для разбиения исходного куба используется интервальная стратегия (см. рис. 2).

Пусть N_j – это число интервалов, на которое разбивается j-е измерение (записи j-й таблицы); $N_j = |i_j|$, i_j – второй индекс в N_{ji_j} (см. рис. 2). Ясно, что

$$\prod_{i=1}^{n} N_i = N, \qquad (5)$$

так как левая часть равенства определяет число областей, на которые разбивается исходный куб S, а оно равно числу узлов в системе N.

По аналогии с (3) получим общее число дублирований кортежей по всем узлам сети для случая интервального разбиения:

$$\Psi(C_I) = \sum_{i=1}^{n}|R_i|\prod_{j\neq i}^{n} N_j = N \cdot \sum_{i=1}^{n}\frac{|R_i|}{N_i}. \qquad (6)$$

Но в соответствии с неравенством Коши (среднеарифметическое не меньше среднегеометрического) имеем:

$$\Psi(C_I) = N \cdot \sum_{i=1}^{n} \frac{|R_i|}{N_i} \geq n \cdot N \cdot \sqrt[n]{\prod_{i=1}^{n} \frac{|R_i|}{N_i}} = n \cdot N \cdot \sqrt[n]{\frac{\prod_{i=1}^{n} |R_i|}{N}} = \Psi(C_H). \quad (7)$$

Доказательство теоремы 1 *завершено*.

Следует отметить, что в работе [1] просто доказывается, что стратегия C_H входит в класс так называемых «идеальных функций секционирования», т.е. функций, минимизирующих $\Psi(C)$.

Как следует из теоремы 1, интервальная стратегия является также оптимальной по критерию $\Psi(C)$ в случае, если $|R_i|/N_i = |R_j|/N_j$ для $\forall i, j$. Это следует из того, что $\Psi(C_H)$ не зависит от $\{N_i\}$ (см. (7)). И поэтому значение $\Psi(C_H)$ для заданных N является нижней границей для $\Psi(C_I)$. Учитывая (5), легко получить, что оптимальное число интервалов N_iопределяется следующим выражением:

$$N_i = |R_i| / \sqrt[n]{\frac{\prod_{i=1}^{n} |R_i|}{N}}. \quad (8)$$

Это значение совпадает с тем, что было получено в [9] путем решения задачи вариационного исчисления.

Таким образом, стратегия разбиения C_H лучше стратегии C_I. Но это касается числа копируемых записей в узлы системы. Однако остается вопрос: справедливо ли это утверждение для объема копируемых данных? Как показано в следующем разделе, ответ на поставленный вопрос не однозначен.

Объем данных, передаваемых по сети с учетом фрагментации таблиц R_i

Записи исходных таблиц хешируются и хранятся в виде фрагментов в локальных файлах узлов. Во многих работах, в частности в [1], эта особенность системы MapReduce не учитывается.

Будем предполагать, что известен номер каждой записи. Вдоль каждой оси (таблицы) записи упорядочены по своим номерам (рис. 3). Номера узлов в общем случае не упорядочены.

1. Стратегия C_H.

На рис. 3 показано, что 2^m записей каждого измерения (таблицы) копируются в j-й узел для обработки (т.е. для тета-соединения). Номер узла, куда пересылаются записи, определяется при движении вдоль кривой Гильберта (см. рис. 1)или любым другим правилом нумерации. Если используется кривая Гильберта, то номер узла увеличивается на 1 после прохождения каждых $\left(2^m\right)^n$ элементов исходного куба S.

В каждый узел предварительно были записаны (в соответствии с фрагментацией) $|R_i|/N$ записей i-й таблицы, i = 1,…, n. Число интервалов q_i, на которое разбивается каждое i-е измерение, определяется выражением (2).

Тогда из каждого из

$$|Y_{ik}| = 2^m / (|R_i|/N) \quad (9)$$

узлов $|R_i|/N$ записей таблицы R_i копируются на

$$|X_{ik}| = \prod_{j \neq i}^{n} q_j \qquad (10)$$

узлов (см «сечение» на рис. 3), где i = 1,…,n; k – номер группы узлов, всего имеется $|R_i|/2^m$ групп узлов по i-му измерению.

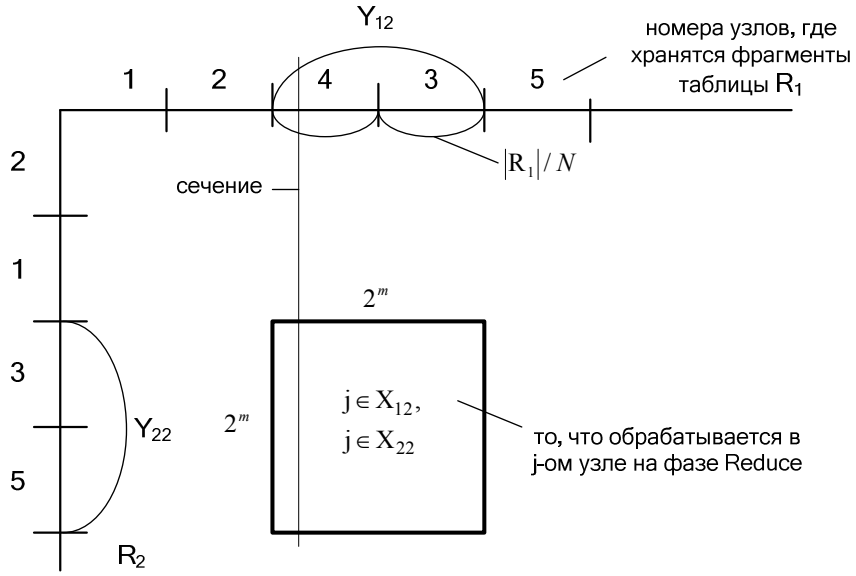


Рис. 3. Дублирование (копирование) записей подкуба $2^m \times 2^m \times \ldots \times 2^m$.

Для разных групп Y_{ik} множества выходных узлов X_{ik} не пересекаются. Имеют место следующие соотношения: $|Y_{ik}| = |X_{ik}|$ (это следует из (1) и (2)), $Y_{ik_1} \cap Y_{ik_2} = \varnothing$, $X_{ik_1} \cap X_{ik_2} = \varnothing$, $\sum_k |Y_{ik}| = \sum_k |X_{ik}| = N$, но в общем случае $Y_{ik} \cap X_{ik} \neq \varnothing$.

Вероятность, что какой-либо исходный узел $j \in Y_{ik}$ попадет в множество X_{ik}, определим, используя классическое определение вероятности:

$$P_i = |X_{ik}|/N = 2^m/|R_i|. \qquad (11)$$

Из каждого исходного узла $j \in Y_{ik}$ по сети копируется в каждый из $|X_{ik}|-1$ узлов $|R_i|/N$ записей. Но в один оставшийся узел (из $|X_{ik}|$) в среднем по сети копируется $(1-P_i)|R_i|/N$ записей. Тогда средний объем данных, передаваемых по сети из одного какого-либо узла, можно рассчитать по формуле:

$$Q_H = \sum_{i=1}^{n}((|X_{ik}|-1)\frac{|R_i|}{N} + (1-P_i)\frac{|R_i|}{N})L_i = \sqrt[n]{\frac{\prod_{i=1}^{n}|R_i|}{N}}(1-\frac{1}{N})\sum_{i=1}^{n} L_i, \qquad (12)$$

где L_i – длина записи таблицы R_i.

Можно проверить, что из сети в один какой-либо узел поступает такой же объем.

2. Стратегия C_I.

По аналогии с предыдущем случаем можно получить средний объем дан-

ных, передаваемых по сети из одного какого-либо узла для стратегии C_I. Заменив 2^m на $|R_i|/N_i$ (см. рис. 3), получим

$$Q_I = (1-\frac{1}{N})\sum_{i=1}^{n}\frac{|R_i|\cdot L_i}{N_i}. \quad (13)$$

Сравним (12) и (13). Представим (12) в следующем виде:

$$Q_H = (1-\frac{1}{N})\sqrt[n]{\prod_{i=1}^{n}\frac{|R_i|\cdot L_i}{N_i}}\cdot n\cdot\frac{\frac{1}{n}\sum_{i=1}^{n}L_i}{\sqrt[n]{\prod_{i=1}^{n}L_i}}. \quad (14)$$

Разделим (14) на (13):

$$\frac{Q_H}{Q_I}=\frac{\sqrt[n]{\prod_{i=1}^{n}B_i}}{\frac{1}{n}\sum_{i=1}^{n}B_i}\times\frac{\frac{1}{n}\sum_{i=1}^{n}L_i}{\sqrt[n]{\prod_{i=1}^{n}L_i}}, B_i=\frac{|R_i|\cdot L_i}{N_i}. \quad (15)$$

В силу неравенства Коши первый сомножитель в (15) не превышает 1, а второй сомножитель больше или равен 1. На основании формулы (15) можно сделать следующие выводы:

1) если $|R_i|/N_i = |R_j|/N_j$ для $\forall i, j$, то выражение (15) будет равно 1;

2) если $B_i = B_j$ для $\forall i, j$, но $\exists i, j$, для которых $L_i \neq L_j$, то стратегия Q_I лучше;

3) если $L_i = L_j$ для $\forall i, j$, но $\exists i, j$, для которых $B_i \neq B_j$, то лучше стратегия Q_H.

Оценка времени выполнения многотабличного тета-соединения

Полученные в [1] оценки стоимости выполнения тета-соединения имеют существенный недостаток: в стоимость входит только составляющие, связанные с передачей данных по сети и вводом/выводом на диск, т.е. не учитывается процессорное время выполнения соединения. Ниже приведены оценки среднего времени выполнения многотабличного тета-соединения по технологии MapReduce, которые не имеют указанных недостатков.

Ниже приведены спецификации функций Map и Reduce, реализующих многотабличное тета-соединение с помощью одного задания (Job).

Map:

Для $(i_1,\ldots,i_n)\in U_{1j}\times\ldots\times U_{nj}$

$$w=\coprod_{k=1}^{n}\left\lceil\frac{i_k}{g_k}\right\rceil \quad (16)$$

Output $< w, R_k(i_k) >, k = 1,\ldots,n$

Конец для

В приведенных выше спецификациях приняты следующие обозначения: U_{kj} – множество номеров кортежей k-й таблицы в j-м узле (номер больше 0); w – номер узла, куда дублируются (копируются) записи таблиц,

$g_k = |R_k|/N_k$ (17)

– для дублирования на основе интервальной стратегии; $|R_k|$ – общее число кортежей в k-й таблице; N_k – число интервалов, на которое разбивается k-е измерение, т.е. записи k-й таблицы, или

$g_k = 2^m$ (18)

– для дублирования на основе кривой Гильберта; <w, $R_k(i_k)$> – запись, помещаемая в выходной поток; $R_k(i_k)$ – i_k кортеж k-й таблицы; ⌈ ⌉ – округление с избытком.

Так как число соединяемых таблиц (n) может быть достаточно большим, то даже при небольшом числе записей $|R_k|$ количество перебираемых комбинаций номеров записей в узле $(i_1,\ldots,i_n)$ может быть очень велико ($\prod_{k=1}^{n} g_k$). Поэтому перед соединением в узле таблицы группируют, далее их соединяют в каждой группе, затем уже соединяют получившиеся результаты [1]. Ниже приведены спецификации функции Reduceдля случая итеративного парного соединения таблиц, так как этот вариант наиболее подходит для маломощных станций, которые используются в системах MapReduce.

Reduce:
H = R_{1w}
Для i = 2,n
  Чтение H
  Для $(h, r_i) \in H \times R_{iw}$
  ЕСЛИ P(h,r_i) = true, ТО записать (h,r_i) в H (19)
  Конец для
Конец для
Чтение H
Output $< k,\ list\, a_{ki}.r_k >, k = 1,\ldots,n$

Здесь Riw – множество кортежей i-й таблицы, поступивших на узел w на этапе shuffle; H – промежуточная таблица; P(·) – предикат, соответствующий условию тета-соединения listaki; r_k – список значений атрибутов кортежа r_k, помещаемых в выходной поток как результат выполнения тета-соединения.

Предположим, что для реализации многотабличного тета-соединения используется сеть на коммутаторах, объединенных в кольцо. В [10] было получено выражение для среднего времени передачи всех промежуточных файлов системы на некоторой фазе выполнения запроса к базе данных. По аналогии можно получить формулу для среднего времени передачи данных между фазами Мари Reduceдля тета-соединения таблиц:

$$T_{RNW}(Q) = Q \max\{\frac{1}{\eta \cdot \mu_{DR}}, \frac{1}{\mu_{PW}}, K\frac{K-1}{N-1}\frac{1}{\mu_{N1}}, N\frac{N-K}{N-1}\frac{1}{\mu_{N2}}, \frac{1}{\mu_{PR}}, \frac{1}{\eta \cdot \mu_{DW}}\}. \quad (20)$$

Здесь приняты следующие обозначения: μ_{DR} – пропускная способность диска (чтение); μ_{PW} – пропускная способность порта коммутатора (выход); μ_{N1} – пропу-

скная способность коммутирующей матрицы коммутатора; μ_{N2} – пропускная способность соединительного кольца (стека); μ_{PR} – пропускная способность порта коммутатора (вход); μ_{DW} – пропускная способность диска (запись); Q – средний объем данных, передаваемых по сети из одного какого-либо узла (определяется выражениями (13) и (14)); $\eta = (1 - 1/N)$ – коэффициент, учитывающий дополнительный объем чтения/записи на диск; K– число узлов подключенных к коммутатору; N – общее число узлов. Пропускные способности ресурсов заданы в байт/с. Коэффициент (K – 1)/(N – 1) определяет вероятность, что данные от K узлов будут переданы в рамках одного коммутатора. Коэффициент (N – K)/(N – 1) определяет вероятность, что данные от N узлов будут переданы в узлы, находящиеся за пределами одного коммутатора (будут переданы через соединительное кольцо, т.е. стек).

Формула (20) определяет время передачи данных через самый загруженный ресурс.

Все узлы одинаковые и работают параллельно при перекрестной передаче данных между узлами после выполнения фазы Map (shuffle).

Среднее время чтения и обработки записей таблиц Ri функцией Map (см. (16)) можно представить в виде следующего выражения:

$$T_M = t_Z + \max\{\frac{V}{\mu_{DR1}} + t_{PR1},\ \frac{Q}{\mu_{DW2}}\}. \qquad (21)$$

Чтение записей и их обработка функцией Map выполняются последовательно, а обработка записей в буфере и запись их в файлы выполняются в фоновом режиме, т.е. параллельно (см. max в формуле). Сортировка записей не требуется.

Поясним обозначения в формуле (21): t_Z – время распространения функций Map и Reduce по узлам системы; V– объем данных, читаемых из файловой системы MapReduce; Q – объем данных, копируемый на другие узлы (определяется выражениями (13) и (14)); t_{PR1} – процессорное время обработки входных записей функцией Map; μ_{DR1} – пропускная способность файловой системы MR на чтение; μ_{DW2} – пропускная способность локального диска на запись.

С учетом спецификаций функции Map (16) получим

$$V = \frac{1}{N}\sum_{i=1}^{n}|R_i|L_i\ . \qquad (22)$$

$$t_{PR1} = \tau \cdot \prod_{i=1}^{n}\frac{|R_i|}{N}\cdot(3n-1), \qquad (23)$$

τ – среднее время выполнения одной короткой логической операции алгоритма (КЛОА); (3n – 1) – число операций, необходимых для вычисления номера узла w; 2n – число делений и округлений с избытком; (n –1) – число умножений.

Среднее время пересылки и обработки записей таблиц R_i функцией Reduce можно представить в виде следующего выражения:

$$T_R = T_{RNW}(Q) + \frac{Q}{\mu_{DR2}} + T_{Rn}, \qquad (24)$$

где T_{RNW} – среднее время передачи всех промежуточных файлов, полученных функциями Map, определяется выражением (20); Q – объем данных, поступивший в узел из других узлов (определяется выражениями (13) и (14)); μ_{DR2} – пропускная способность локального диска на чтение; T_{Rn} определяется с помощью следующей рекуррентной формулы (см. спецификации (19)):

$$T_{Ri} = T_{R\,i-1} + \max\{\frac{H_{i-1}}{\mu_{DR2}} + t_{PR2i},\ \frac{H_i}{\mu_{DW2}}\}, \qquad (25)$$

$i = 2,\ldots,n,\ T_{R1} = H_1 = 0.$

Чтение записей, полученных на предыдущем шаге (H_{i-1}), и их обработка в процессоре на текущем шаге (t_{PR2i}) выполняются последовательно, а сохранение результатов выполнения текущего шага тета-соединения на локальном диске выполняется в фоновом режиме, т.е. параллельно (см. max в формуле).

Поясним обозначения в формуле (25): T_{Ri-1} – время реализации соединения в узле (i – 1) таблиц; Hi_{-1} – объем промежуточной таблицы соединения (i – 1) таблиц; t_{PR2i} – процессорное время соединения промежуточной таблицы, полученной на предыдущем шаге, с i-й таблицей; H_i – объем результата соединения на i-м шаге; μ_{DR2}, μ_{DW2} – пропускные способности локального диска на чтение и запись.

$$H_i = F_i \sum_{j=1}^{i} L_j, \qquad (26)$$

где F_i – оценка числа записей после соединения iисходных таблиц в узле, определяется следующей рекуррентной формулой:

$$F_i = F_{i-1} g_i p_1 p_2^{i-2}, \qquad (27)$$

$F_1 = g_1,$

здесь g_i определяется формулой (17) или (18); вероятность p_1 учитывает селективность условия соединения i-й таблицы с (i – 1)-й таблицей; вероятность p_2 учитывает селективность условий соединения i-й таблицы с остальными (i – 2) таблицами.

$$t_{PR2i} = \tau \cdot F_{i-1} \cdot g_i \cdot (o_i + l_i + z_i), \qquad (28)$$

τ – среднее время выполнения одной КЛОА; o_i – число арифметических операций сравнений (=, <>, >, <, iни др.) в условии тета-соединения на i-м шаге; l_i – число логических операций; z_i – число перемещений указателей на значения атрибутов.

На последнем n-м шаге выражение H_n/μ_{DW2} в формуле (25) следует заменить на следующее выражение, учитывающее запись результата в файловую систему MapReduce:

$$F_n \cdot \sum_{i=1}^{n} L_{ai}/\mu_{DW1}, \qquad (29)$$

где μ_{DW1} – пропускная способность файловой системы MR на запись; L_{ai} – длина кортежа проекции таблицы R_i на множество атрибутов, указанных в качестве результата выполнения тета-соединения.

Суммарное время выполнения тета-соединения можно оценить по формуле:

$$T = T_M + T_R, \qquad (30)$$

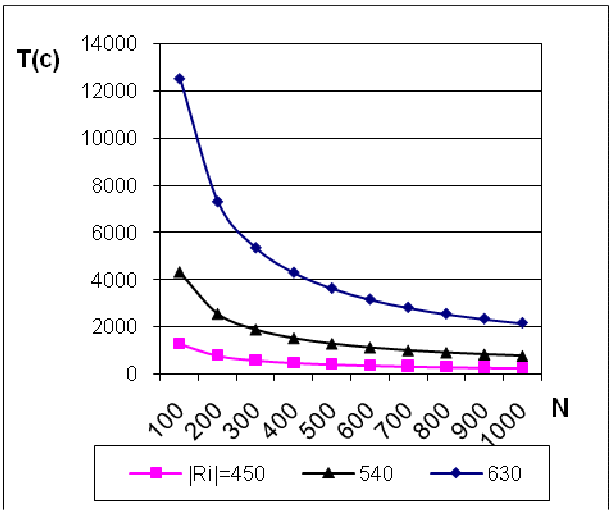
где T_M иT_R определяются выражениями (21) и (24).

Пример расчета времени тета-соединения

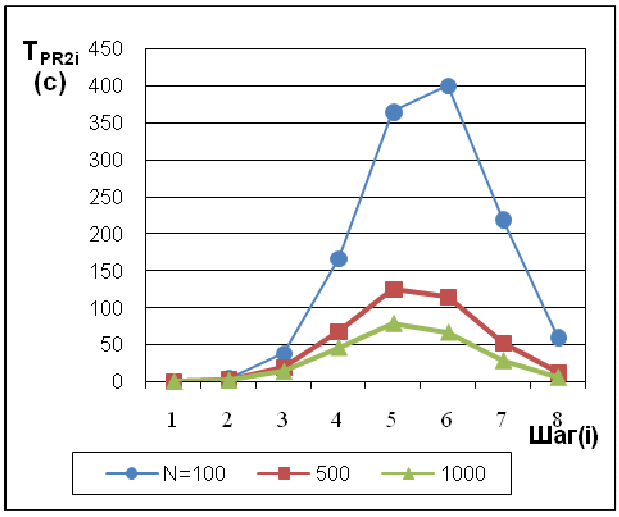
Рассмотрим пример запроса к базе данных, приведенный во введении (см. оператор SELECT). Анализируются маршруты между 10 городами, между каждой парой городов выполняются «k» рейсов ($|R_i| = 10 \cdot 9 \cdot k$). В таблице приведены исходные данные, используемые при расчетах.

Параметр	Значение	Параметр	Значение
n – число таблиц в соединении	9	K (число портов коммутатора)	50
$\|R_i\| = \|R\|$ - число записей в i-й таблице	90k	μ_{DR1} – HDFS на чтение (21)	35 Мбайт/с
L_i – длина записи таблицы	48	μ_{DW1} – HDFS на запись (29)	8 Мбайт/с
$2^m = \sqrt[n]{\frac{\prod_{j=1}^{n}\|R_j\|}{N}}$	$\frac{\|R\|}{\sqrt[n]{N}}$	μ_{PW}, производительности в (20) μ_{N1}, μ_{N2}, μ_{PR}	1 Гбит/с, 128 Гбит/с, 64 Гбит/с, 1 Гбит/с
$N_i = \sqrt[n]{N}$ – число интервалов	$\sqrt[n]{N}$	μ_{DR}, μ_{DR2} – SATA1 на чтение (20), (24), (25)	50 Мбайт/с
L_{ai} – длина кортежа проекции	4	μ_{DW}, μ_{DW2} – SATA1 на запись (20), (25)	50 Мбайт/с
$o_i + l_i + z_i$ (i = 2,…,n)	(6+i)+(4+i)+188	t_Z – время распространения	10 с
p_1 – вероятность успеха 1	1/16	τ – время КЛОА [11]	$1{,}6 \cdot 10^{-8}$ с
p_2 – вероятность успеха 2	1/2		

При расчетах использовалась формула (30). На рис. 4а приведен график зависимости времени выполнения тета-соединения от числа параллельно работающих узлов в системе MapReduce. В рассматриваемом примере обе стратегии Q_H и Q_I показывают одинаковый результат в силу заданных исходных данных. Как видно из графиков, время существенно зависит от числа обрабатываемых записей в таблице FI. Даже при N = 1000 время обработки велико и для $|R_i| = 450$, T = 253 с, для $|R_i| = 540$, T = 787 с, для $|R_i| = 630$, T = 2161 с, где число записей $|R_i| = |s_i|$ (i=1,…, n = 9) соответствует числу рейсов k = 5,6,7 для каждой пары городов.



а



б

Рис. 4. Результаты вычислительных экспериментов.

Заключение

Рассмотрены две стратегии дублирования (копирования) записей таблиц на узлы системы MapReduce с целью сегментации исходного n-мерного куба кортежей – на основе кривых Гильберта и интервальная стратегия. Рассмотрены варианты преимущественного использования каждой стратегии, дающие возможность уменьшить объем передаваемых по сети данных (см. (15)). Получены выражения для оценки времени выполнения многотабличного тета-соединения (формулы (20) – (30)), позволяющие учитывать процессорную, дисковую и сетевую составляющие. Рассмотрен практический пример, показывающий, что «узким местом» системы может стать процессор маломощных станций, которые используются в узлах каркаса MapReduce.

ЛИТЕРАТУРА

1. *Zhang X., Chen L., Wang M.* Efficient multi-way theta-join processing using MapReduce // PVLDB. – 2012. – 5(11). – P.1184-1195.
2. *Chaudhuri S. and et al*. Optimization of real conjunctive queries // PODS. – 1993. –P.59-70.
3. *Tan K.-L. and et al.* A note on the strategy space of multiway join query optimization problem in parallel systems // SIGMOD Record. – 1991. – 20(4). – P.81-82.
4. *Lee C. and et al.* Optimizing large join queries using a graph-based approach // TKDE. – 2001. – 13(2). – P.298-315.
5. *Borthakur D. and et al.* Apache hadoop goes realtime at facebook // SIGMOD. – 2011. – P. 1071-1080.
6. *Das S. and et al.* G-store: a scalable data store for transactional multi key access in the cloud // SoCC. – 2010. – P.163-174.
7. *Lawder J.K.* Calculation of Mappings Between One and n-dimensional Values Using the Hilbert Space-filling Curve. Technical Reportno. JL1/00, August15, 2000. [http://www.learninglink.bbk.ac.uk/research/techreps/2000/bbkcs-00-01.pdf].
8. *Arthur R.Butz.* Alternative Algorithm for Hilbert's Space-Filling Curve. // IEEE Transactionson Computers. – 1971. – 20. – P.424-426.
9. *Zhang C., Li J., Wu L.* Optimizing Theta-Joins in a MapReduce Environment // International Journal of Database Theory and Application. – 2013. – Vol. 6, No. 4. – P.91-107.
10. *Григорьев Ю.А., Плутенко А.Д.* Анализ процесса выполнения запроса на соединение таблиц в строчной параллельной СУБД // Информатика и системы управления. – 2013. – № 4. – С.3-16.
11. Григорьев Ю.А., Плутенко А.Д. Анализ времени соединения таблиц в строчной параллельной системе баз данных и по технологии MapReduce // Информатика и системы управления. – 2014. –№ 2. – С.3-11.

E-mail:
Григорьев Юрий Александрович – grigorev@bmstu.ru;
Плутенко Андрей Долиевич – plutenko@bk.ru.