УДК 004.657

© 2014 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)

ОЦЕНКА ВРЕМЕНИ СОЕДИНЕНИЯ ТАБЛИЦ В БАЗЕ ДАННЫХ NOSQL ПО ТЕХНОЛОГИИ MAPREDUCE

Получены выражения времени выполнения запроса на соединение таблиц по технологии MapReduce (MR). Выбор задачи соединения был связан с тем, что в рамках этого одного запроса можно реализовать все основные операции поиска данных: селекцию, соединение, агрегацию, проекцию, группирование и сортировку. Для оценки процессорной составляющей в модель введен параметр времени выполнения короткой логической операции алгоритма (КЛОА), позволяющий выполнять калибровку модели по результатам натурных экспериментов. Модель учитывает большое число параметров, влияющих на время выполнения запроса, в частности число узлов в кластере, объем базы данных в узле, селективность атрибута в условии поиска, число сортируемых записей, мощность атрибута группирования и др.
Ключевые слова: база данных NoSQL, технология MapReduce, запрос на соединение таблиц, оценка времени выполнения запроса.

Введение

В работе [1] утверждается, что при использовании баз данных NoSQL «требуемые характеристики масштабируемости и отказоустойчивости может обеспечить технология MapReduce (MR), поскольку она с самого начала разрабатывалась с расчетом на масштабирование до тысяч узлов, и ее реализация от Google эффективно используется для поддержки внутренних операций этой компании». Изначально технология MapReduce ориентировалась на обработку огромных объемов неструктурированных текстовых данных и в настоящее время находит все большее применение [1].

Однако в статье [2] на конкретных тестовых примерах показано, что при обработке структурированных данных MapReduce не может конкурировать по производительности с параллельными СУБД. При этом из публикации не ясно, всегда ли технология MapReduce проигрывает параллельным СУБД или только при некоторых условиях, как скажется на производительности параллельных баз данных увеличение объема данных, хранимых в узлах кластера?

В работе [3] разработана аналитическая модель соединения таблиц в строчной параллельной СУБД. Выбор задачи соединения был связан с тем, что в рамках этого одного запроса можно реализовать все основные операции SQL: селекцию (selection), соединение (join), агрегацию (aggregation), проекцию (projection), группирование (grouping) и сортировку (sorting). Запрос на соединение таблиц – наиболее ресурсоемкая операция в базах данных. В данной статье получено среднее время выполнения этого запроса в базе данных NoSQL (по технологии MapReduce). В следующей публикации будет выполнено сравнение времени выполнения соединения таблиц в двух средах: в строчной параллельной СУБД и по технологии MapReduce.

Технология MapReduce

В системе NoSQL данные хранятся в виде записей (ключ, значение). Значения можно задавать, используя различные форматы: текст, html (потом это значение можно посмотреть, используя браузер), json («имя поля»: «значение поля»), XML и др.

В БД NoSQL параллелизм доступа к данным обеспечивается путем применения технологии MapReduce [1, 2]. Сначала данные фрагментируются по узлам системы. Записи распределяются по узлам. В качестве узлов выступают компьютеры, объединенные в кластер.

Запрос к базе данных NoSQL по технологии MapReduce включает описание двух функций: функции, выполняемой на фазе Map, и функции, выполняемой на фазе Reduce. Рассмотрим, как выполняется запрос в базе данных NoSQL Hadoop [2, 4].

Система автоматически тиражирует функцию «map» по узлам кластера. На каждом узле может быть запущено несколько экземпляров «map». Далее система инициирует чтение записей $\{(k_{ji}, v_{ji})\}$ из базы данных в каждом j-м узле и последовательную передачу их на вход функции «map» ($j = 1, ..., n$). Эти функции выполняются параллельно на разных узлах кластера. Программу «map» (как и программу «reduce») разрабатывает программист под конкретную задачу. Эта функция должна выполнить следующие действия: разобрать входную запись и вернуть новую запись (m_{ji}, r_{ji}). При этом в отличие от входных записей значение ключа m_{ji} может повторяться. Записи хешируются по значению ключа m_{ji}, формируются разделы записей, номер раздела совпадает со значением хеш-функции. Число разделов обозначим через s. Система объединяет выходные записи программы «map» в s массивах и сохраняет их на локальном диске узла (один массив на одно значение хеш-ключа). Таким образом, формируются $n \cdot s$ файлов, где n – число узлов.

После завершения фазы Map каждый результирующий массив F_{ji} записей ($j = 1, ..., n$, $i = 1, ..., s$) пересылается с j-го узла на узел, где выполняется i-й экземпляр функции Reduce. Важно подчеркнуть, что этот механизм пересылки массивов позволяет обрабатывать записи с одинаковыми значениями ключа в одном узле. Поступившие в узел записи массивов сортируются, объединяются и передаются на вход соответствующей функции «reduce». Программа «reduce» должна

обработать записи объединенного массива и выполнить их «свертку» в соответствии с поставленной задачей. Функция «reduce» возвращает выходной массив, как правило, меньшей размерности, и система сохраняет его в виде записей (ключ, значение) в файловой системе базы данных NoSQL. Эти записи могут быть использованы в качестве входных данных для других запросов к БД.

Следует отметить, что программист разрабатывает процедуры «map» и «reduce» (т.е. формирует запрос), а тиражирование их по узлам, параллельное выполнение экземпляров программ, синхронизацию фаз, сортировку и объединение выходных массивов, передачу записей и массивов на вход процедурам обеспечивает сама система MapReduce. При этом на разных узлах и уровнях выполняется одна и та же программа «map» и программа «reduce», и это необходимо учитывать при разработке указанных процедур. В одном узле может выполняться несколько экземпляров функции «map».

Оценка среднего времени соединения двух таблиц в базе данных NoSQL по технологии MapReduce

Ниже приведен типичный SQL-запрос на соединение двух таблиц R1 и R2:

```
SELECT R1.A1X, f1(атр1) as agr1, f2(атр2) as agr2, ..., fq(атрq) as agrq
FROM R1, R2
WHERE R1.A11 = R2.A21 AND условие по R1 AND условие по R2          (1)
GROUP BY  R1.A1X
ORDER BY agr1 [DESC] [LIMIT  Y];
```

здесь f1, f2, ... – некоторые агрегатные функции (AVG, SUM и др.).

MR-программу, реализующую задачу соединения (1), приходится разбивать на три разные фазы [2]. Все эти фазы реализуются вместе, как одна MR-программа в NoSQL, но следующая фаза не начинает выполняться, пока не завершится предыдущая. На рис. 1 приведена схема функционирования MR на какой-либо фазе.

Фаза 1. На первой фазе отсеиваются записи таблиц R1 и R2, которые не удовлетворяют условиям поиска, и оставшиеся записи соединяются.

Функция Map. На каждом узле запускаются L экземпляров функции Map (метка 2, метки на рис. 1 подчеркнуты), которые читают из файловой системы базы данных NoSQL записи таблиц R1 и R2 в виде пары «ключ/значение» (метка 1).

Значение расщепляется на поля и к записям таблиц применяются фильтры «условие для R1» и «условие для R2» (см. запрос (1)). Записи, удовлетворяющие условию, сначала выводятся в буфер памяти с составным ключом (метка 3). Записи R1 выводятся с составным ключом (A11, 1), где A11 – значение атрибута соединения; 1 – признак записи R1. Записи R2 выводятся с составным ключом (A21, 2), где A21 – значение атрибута соединения; 2 – признак записи R2. В дальнейшем первую часть (A11 или A21) составного ключа будем называть просто ключом.

Каждой функции Map выделяется буфер памяти объемом (по умолчанию он равен 100 Мбайт) [9]. Когда объем заполнения буфера превышает определенный

порог (по умолчанию он равен 80%), то запускается отдельная задача, которая выполняется в фоновом режиме. Она разбивает буфер на разделы и сортирует каждый раздел по ключу (in-memory sort). После этого каждый раздел буфера выводится на диске в виде файла (spill to disk) (метка 4). Запись принадлежит *i*-му разделу, если h(A11 или A21) = *i*, где *h* – некоторая хеш-функция. Обычно число разделов s равно числу узлов в кластере, т.е. $s = n$.

Таким образом, общее число отсортированных файлов, которые будут сохранены на диске в процессе выполнения всех функций Map-узла, будет равно $r \cdot s \cdot L$, здесь

$$r = V/(V_B \cdot P_T), \qquad (2)$$

где V – общий объем выходных данных, сформированных одним экземпляром функции Map; V_B – размер буфера памяти; P_T – порог заполнения буфера; L – число экземпляров функции Map, запущенных на одном узле.

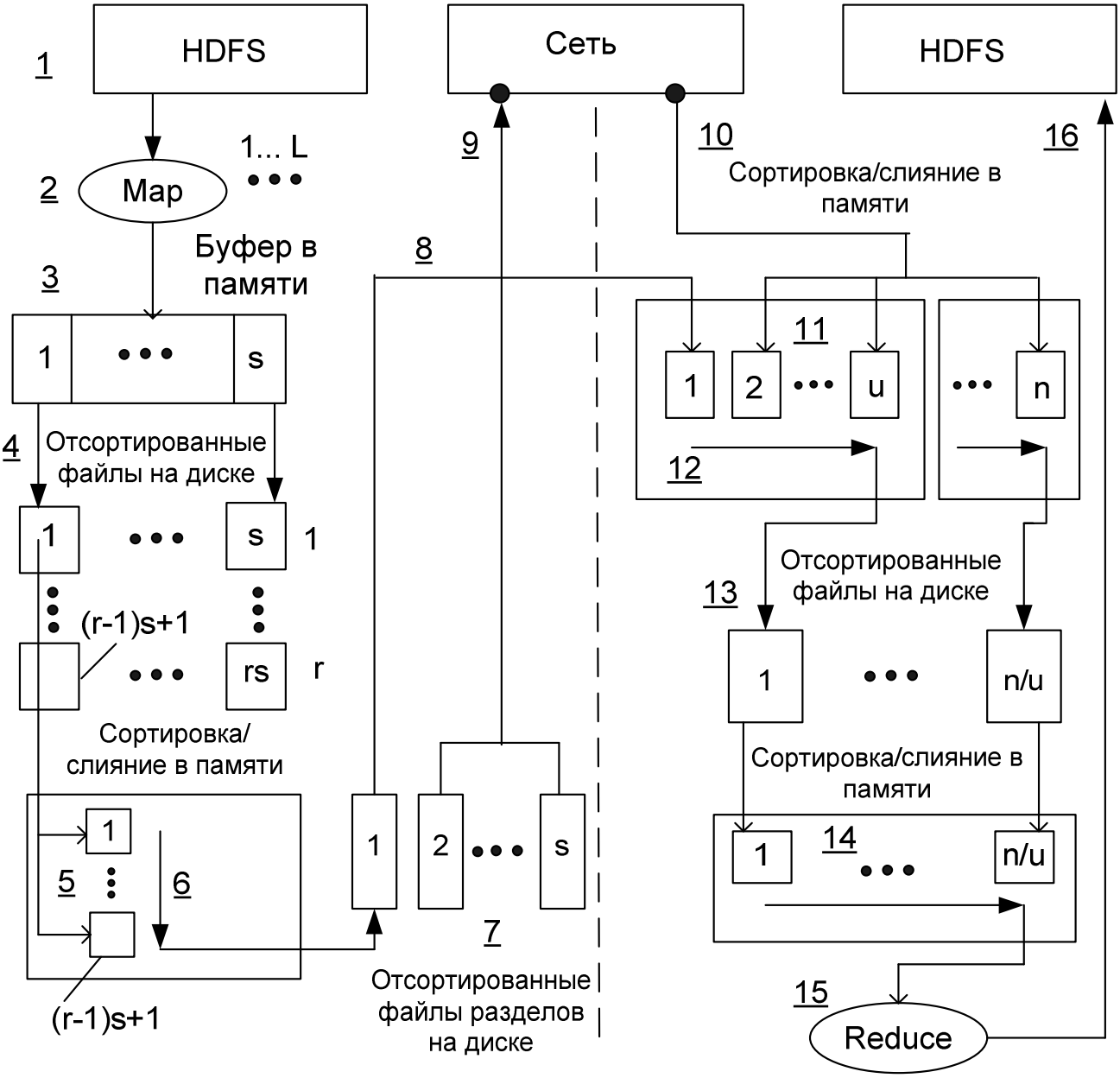

Рис. 1. Схема выполнения соединения таблиц по технологии MapReduce.

Примечание. Если задействована возможность Combiner (по умолчанию она отключена), то сначала каждый раздел, сформированный в буфере памяти, подается на вход функции Reduce, и только потом результат выполнения функции Reduce сохраняется на диске (см. метку 4) [5]. Для некоторых задач это позволяет уменьшить объем обрабатываемых данных. Но для рассматриваемого случая соединения таблиц эта возможность должна быть отключена.

После того, как функции Map обработают все входные записи, в узле запускается специальная процедура. Она сортирует записи каждого раздела $i = 1, ..., s$ и объединяет их в один файл (merge on disk). Для этого процедура выделяет в памяти $r \cdot L$ блоков (буферов) – по одному на каждый файл i-го раздела (метка 5). И читает записи из i-го файла, принадлежащего i-му разделу, в 1-й блок, из $(s + i)$-го файла – во 2-й блок и т.д. В каждом блоке записи уже отсортированы на предыдущем этапе. Поэтому сравниваются первые записи этих блоков по ключу ($r \cdot L$ сравнений). Запись с минимальным значением ключа перемещается в буфер диска (одно перемещение) (метка 6). Остальные записи соответствующего блока сдвигаются вправо (блок работает как стек). Затем снова сравниваются первые записи $r \cdot L$ блоков по ключу и т.д. Если записи в каком-либо блоке исчерпаны, то в этот блок подгружаются записи из связанного с ним файла. После обработки таким способом $r \cdot L$ файлов будет сформирован отсортированный файл i-го раздела. В результате выполнения указанной процедуры будут созданы s отсортированных файлов – один на каждый раздел (метка 7).

Априори оценить процессорную составляющую описанных выше и ниже процессов соединения таблиц непросто. По аналогии с [3] можно предложить следующий метод. Зная алгоритм выполнения процедуры, можно подсчитать число коротких логических операций алгоритма (КЛОА), а затем с помощью калибровки модели определить процессорное время одной такой операции.

Ниже используются следующие обозначения: n – число узлов в кластере; s – число разделов ($s = n$); τ – среднее время выполнения одной КЛОА; t_Z – время распространения функций Map и Reduce по узлам системы; V_1, V_2 – объемы таблиц R1 и R2, хранящихся в одном узле системы; Q_i – число записей в таблице Ri ($i = 1, 2$); L – число функций Map, запускаемых на одном узле; R1P, R2P – проекции исходных таблиц R1 и R2 на множество атрибутов R1.A11, R2.A21, R1.A1X, атр1, атр2, ..., атрq (см. запрос (1)); V_{1P}, V_{2P} – объемы этих проекций; V_J – объем записей, полученных после соединения проекций R1P и R2P (см. (18)); Q_J – число записей в соединении, выполненном на одном узле; V_{AY1} – объем записей, получаемых после агрегирования значений атрибутов и сохраняемых MR в базе данных NoSQL; p_1 и p_2 – доли записей таблиц R1 и R2, удовлетворяющих условиям поиска; T_{RNW} – среднее время передачи всех промежуточных файлов системы на какой-либо фазе выполнения соединения таблиц, определяется выражением (1) в [3]; T_{RNW3} – среднее время передачи всех промежуточных файлов системы на 3-й фазе (см. (33)), выводится по аналогии с (1) в [3]; μ_{DR1} – пропускная способность файловой системы базы данных NoSQL на чтение (например, HDFS СУБД Hadoop); μ_{DR2}, μ_{DR3}, μ_{DR4} – пропускные способности локальных дисков на чтение; μ_{DW2}, μ_{DW3}, μ_{DW4} – пропускные способности локальных дисков на запись; μ_{DW1} –

пропускная способность файловой системы базы данных NoSQL на запись (например, HDFS СУБД Hadoop).

Среднее время чтения и обработки записей таблиц R1 и R2 функциями Map и их сортировки/объединения можно представить в виде следующего выражения:

$$T_{M1} = t_Z + \\ +\max\{(\frac{V_1 + V_2}{L}\frac{1}{\mu_{DR1}} + t_{PR11})L,\ \frac{p_1V_{1P} + p_2V_{2P}}{\mu_{DW2}} + t_{PR12}\} + \\ +\max\{\frac{p_1V_{1P} + p_2V_{2P}}{\mu_{DR2}} + t_{PR13},\ \frac{p_1V_{1P} + p_2V_{2P}}{\mu_{DW3}}\}. \quad (3)$$

Чтение записей и их обработка функцией Map выполняются последовательно (метки 1 и 2 на рис. 1), а обработка записей в буфере и запись их в файлы выполняются в фоновом режиме, т.е. параллельно (метки 3 и 4) – первый max. Аналогично чтение записей разделов и их обработка (сортировка) выполняются последовательно (метки 5 и 6), а их запись в выходные файлы – параллельно (метка 7) – второй max.

Поясним обозначения в формуле (3): t_{PR11} – процессорное время обработки входных записей функцией Map; t_{PR12} – процессорное время сортировки записей в буфере памяти (sort); t_{PR13} – процессорное время сортировки/слияния записей каждого раздела.

Оценим процессорные составляющие на основе числа коротких логических операций алгоритма (КЛОА).

1. $t_{PR11} = \tau(\frac{Q_1}{L}\gamma_{11} + \frac{Q_2}{L}\gamma_{12})$ – метка 2, (4)

$$\gamma_{1i} = 1 + 2V_i/Q_i + 2m_i + o_i + p_i(3 + 3a_i) + p_i, \quad (5)$$

где γ_{1i} – число КЛОА, выполняемых функцией Map при анализе записи таблицы Ri; m_i, o_i – число атрибутов и логических операций в условии поиска в таблице R_i; a_i – число атрибутов в проекции записи таблицы R_i.

2. $t_{PR12} = \tau(r_1\gamma_{21} + r_2\gamma_{22})sL$ – метка 3, (6)

где r_i – количество буферов памяти, разделы которых помещены на диск в процессе выполнения функции Map при обработке записей таблицы Ri, определяется выражением (2) в виде

$$r_i = p_iV_{iP}/L/(V_B \cdot P_T), \quad (7)$$

где γ_{2i} – число КЛОА, выполненных при сортировке

$$bi = p_iQ_i/L/r_i/s, \quad (8)$$

записей в памяти узла. Для сортировки методом слияния (merge sort) [7] величину γ_{2i} можно оценить по формуле

$$\gamma_{2i} = (1+1+1+0.5+1+1+1)b_i\log_2 b_i = 6.5b_i\log_2 b_i. \quad (9)$$

3. $t_{PR13} = \tau(p_1Q_1 + p_2Q_2)\gamma_3$ – метка 6, (10)

γ_3 – число КЛОА, выполненных в процессе слияния промежуточных файлов (см. метку 4), на одну запись. Эту величину можно оценить по формуле

$$\gamma_3 = 1 + ((r_1 + r_2)L - 1)(1 + 1 + 0.5) + 1 + 1 + 1 = 2.5((r_1 + r_2)L - 1) + 4. \quad (11)$$

Функция Reduce. После завершения выполнения всех функций Map система

MR дает команду, узел открывает файлы, подготовленные Map (см. метку 7 на рис.1), читает их и передает записи узлам, где выполняются функции Reduce (метки 8, 9, 10). Файл F_{ji} (j = 1,..., n, i = 1,..., s) пересылается с j-го узла на узел, где выполняется i-й экземпляр функции Reduce. Таким образом, записи с одинаковыми значениями ключа обязательно будут обработаны одной и той же функцией Reduce. Узел принимает записи, сохраняет их в файлах (метка 11) и объединяет их в один отсортированный массив. Для этого исходные файлы группируются по «u» файлов в группе. Схема сортировки/слияния файлов в группе (метка 12) аналогична той, которая была рассмотрена ранее (см. метку 6). Полученные файлы (метка 13) постепенно читаются и сортируются/сливаются в памяти (метка 14). Образуется один отсортированный поток записей (по значениям полей R1.A11 и R2.A21), который без промежуточного запоминания на диске передается на вход функции Reduce.

Функция Reduce выполняет соединение записей с ключами (A11, 1), (A21, 2) по равенству значений полей A11 и A21 (метка 15). Метод соединения совпадает с методом соединения SMJ, который используется в реляционных базах данных (записи во входном потоке уже отсортированы). Функция Reduce выводит в файловую систему базы данных новые пары «ключ/значение»: A1X/(атр1, атр2, ..., атрq) – см. запрос (1) (метка 16).

Среднее время пересылки, сортировки и обработки записей таблиц R1 и R2 функциями Reduce можно представить в виде следующего выражения:

$$T_{R1} = T_{RNW}(p_1V_{1P} + p_2V_{2P}) + \\ +\max\{\frac{p_1V_{1P} + p_2V_{2P}}{\mu_{DR3}} + t_{PR14},\ \frac{p_1V_{1P} + p_2V_{2P}}{\mu_{DW4}}\} + \\ +\max\{\frac{p_1V_{1P} + p_2V_{2P}}{\mu_{DR4}} + t_{PR15},\ \frac{V_J}{\mu_{DW1}}\}. \qquad (12)$$

Чтение записей и их сортировка в процессоре (t_{PR14}) выполняются последовательно (метка 12 на рис. 1), а сохранение записей в файлах выполняется в фоновом режиме, т.е. параллельно (метка 13) – первый max. Аналогично чтение записей, их окончательная сортировка и обработка записей функцией Reduce в процессоре (t_{PR15}) выполняются последовательно (метки 14, 15), а запись результатов соединения записей в базу данных NoSQL – параллельно (метка 16) – второй max.

Поясним обозначения в формуле (12): t_{PR14} – процессорное время сортировки/слияния записей в n/u группах входных файлов, t_{PR15} – процессорное время окончательной сортировки/слияния записей и их соединения при выполнении функции Reduce.

Оценим процессорные составляющие на основе числа коротких логических операций алгоритма (КЛОА).

$$1.\ t_{PR14} = \tau\frac{(p_1Q_1 + p_2Q_2)}{n/u}\gamma_4 \text{ – метка 12}, \qquad (13)$$

где u – число входных файлов в группе; γ_4 – число КЛОА, выполненных в процессе слияния $u > 1$ промежуточных файлов в n/u группах (см. метку 12). Данную величину можно оценить по формуле

$$\gamma_4 = \frac{n}{u}(2.5(u-1)+4), \qquad (14)$$

где *n*/*u* определяет число групп файлов; 2.5(u – 1) + 4 – число КЛОА, выполняемых в процессе слияния одной группы файлов (на одну запись), это число можно вывести по аналогии с выражением (11).

2. $t_{PR15} = \tau(p_1Q_1 + p_2Q_2)\gamma_5 + \tau\gamma_6$ – метки 14, 15, (15)

где γ_5 – число КЛОА, выполненных в процессе слияния *n*/*u* промежуточных файлов (см. метку 14) в один отсортированный поток (на одну запись); по аналогии с (11) эту величину можно оценить по формуле

$$\gamma_5 = 2.5\left(\frac{n}{u}-1\right)+4, \qquad (16)$$

где γ_6 – число КЛОА, выполненных функцией Reduce в процессе соединения записей с ключами (A11, 1) и (A21, 2) по равенству значений полей A11 и A21. Эту величину можно рассчитать по формуле:

$$\begin{aligned}\gamma_6 &= I_1\left(4\frac{p_1Q_1}{I_1} + 4\frac{p_2Q_2}{I_2} + 6\frac{p_1Q_1p_2Q_2}{I_1I_2}\right) + (I_2 - I_1)4\frac{p_2Q_2}{I_2} = \\ &= 4(p_1Q_1 + p_2Q_2) + 6\frac{p_1Q_1p_2Q_2}{I_2},\end{aligned} \qquad (17)$$

где I_1 и I_2 – мощности атрибутов соединения A11 и A21 (число разных значений атрибутов, участвующих в соединении). Без потери общности будем считать, что $I_2 \geq I_1$.

Ниже приведено выражение для оценки объема записей, получаемых после соединения на одном узле и сохраняемых MR в базе данных NoSQL:

$$V_J = l_J \cdot Q_J = l_J \cdot \frac{p_1Q_1p_2Q_2}{I_2}. \qquad (18)$$

Этот объем равен произведению длины новой записи «ключ/значение»: 1X/(атр1, атр2, ..., атрq) на число записей в соединении, выполненном на одном узле.

Можно заметить (см. рис. 1), что в MapReduce задействован мощный механизм сортировки записей. Сортировка выполняется на 4-х уровнях (см. метки 3, 5, 10, 13) в оперативной памяти узлов кластера.

Фаза 2. На этой фазе вычисляются агрегатные значения атрибутов атр1, ..., атрq (см. запрос (1)) на основе значения ключа A1X записей, сгенерированных на фазе 1.

На второй фазе функция Reduce используется для того, чтобы собрать в одном узле все записи с одним и тем же значением A1X.

Функция Map. На каждом узле запускается функция Map (метка 2 на рис. 1), которая принимает записи «ключ/значение» A1X/(атр1, атр2, ..., атрq), сохраненные в базе данных на 1-й фазе, и просто их возвращает как значение функции.

Среднее время чтения записей из базы данных и их сортировки/объединения можно представить в виде следующего выражения:

$$T_{M2} = t_Z + \\ + \max\{(\frac{V_J}{\mu_{DR1}},\ \frac{V_J}{\mu_{DW2}} + t_{PR22}\} + \max\{\frac{V_J}{\mu_{DR2}} + t_{PR23},\ \frac{V_J}{\mu_{DW3}}\}. \quad (19)$$

Формула (19) аналогична выражению (3). Чтение записей и их обработка функцией Map выполняются последовательно (метки 1 и 2 на рис. 1), а обработка записей в буфере и запись их в файлы осуществляются в фоновом режиме, т.е. параллельно (метки 3 и 4) – первый max. Аналогично чтение записей разделов и их обработка (сортировка) выполняются последовательно (метки 5 и 6), а их запись в выходные файлы – параллельно (метка 7) – второй max.

Поясним обозначения в формуле (19): t_{PR22} – процессорное время сортировки записей в буфере памяти (sort); t_{PR23} – процессорное время сортировки/слияния записей каждого раздела.

Эти процессорные составляющие на основе числа коротких логических операций алгоритма оцениваются по аналогии с (6) и (10):

$$1.\ t_{PR22} = \tau \cdot r_3 \cdot \gamma_7 \cdot s \text{ – метка } 3, \quad (20)$$

где τ – среднее время выполнения одной КЛОА; s – число разделов ($s = n$); r_3 – количество буферов памяти, разделы которых помещаются на диск в процессе выполнения функции Map при обработке входных записей таблицы соединения, определяется выражением (2) в виде

$$r_3 = V_J/(V_B \cdot P_T), \quad (21)$$

где γ_7 – число КЛОА, выполняемых при сортировке

$$b_3 = Q_J/r_3/s \quad (22)$$

записей в памяти узла, Q_J определено в выражении (18) . Для сортировки методом слияния (merge sort) величину γ_7 можно оценить по формуле

$$\gamma_7 = 6.5 b_3 \log_2 b_3 . \quad (23)$$

$$2.\ t_{PR23} = \tau \cdot Q_J \cdot \gamma_8 \text{ – метка } 6, \quad (24)$$

где γ_8 – число КЛОА, выполненных в процессе слияния промежуточных файлов (см. метку 4). Эту величину можно оценить по формуле

$$\gamma_8 = 2.5(r_3 - 1) + 4. \quad (25)$$

Функция Reduce. Для каждого значения атрибута A1X эта функция выполняет агрегирование атрибутов атр1, атр2, ..., атрq (см. запрос (1)) – метка 15 на рис. 1. Каждый экземпляр выводит записи с ключом agr1 и значением – кортежем (A1X, agr2, ..., agrq) – метка 16.

Среднее время пересылки, сортировки и обработки записей таблицы соединения функциями Reduce можно представить в виде следующего выражения (по аналогии с (12)):

$$T_{R2} = T_{RNW}(V_J) + \\ + \max\{\frac{V_J}{\mu_{DR3}} + t_{PR24},\ \frac{V_J}{\mu_{DW4}}\} + \max\{\frac{V_J}{\mu_{DR4}} + t_{PR25},\ \frac{V_{AY1}}{\mu_{DW1}}\}. \quad (26)$$

Чтение записей и их сортировка в процессоре (t_{PR24}) выполняются последовательно (метка 12 на рис. 1), а сохранение записей в файлах – в фоновом режиме, т.е. параллельно (метка 13) – первый max. Аналогично чтение записей, их оконча-

тельная сортировка и обработка записей функцией Reduce в процессоре (t_{PR25}) осуществляются последовательно (метки 14, 15), а запись результатов соединения записей в базу данных NoSQL – параллельно (метка 16) – второй max. В формуле (26) t_{PR24} – процессорное время сортировки/слияния записей в *n/u* группах входных файлов; t_{PR25} – процессорное время окончательной сортировки/слияния записей и агрегирования атрибутов при выполнении функции Reduce.

Оценим процессорные составляющие на основе числа коротких логических операций алгоритма (КЛОА) по аналогии с выражениями (13) – (16).

$$1.\ t_{PR24} = \tau \frac{Q_J}{n/u} \gamma_4 \text{ – метка } 12, \qquad (27)$$

где γ_4 – число КЛОА, выполненных в процессе слияния $u > 1$ промежуточных файлов в n/u группах (см. метку 12), определяется выражением (14).

$$2.\ t_{PR25} = \tau Q_J \gamma_5 + \tau \gamma_9 \text{ – метки } 14, 15, \qquad (28)$$

где γ_5 – число КЛОА, выполненных в процессе слияния n/u промежуточных файлов (см. метку 14) в один отсортированный поток (на одну запись), определяется выражением (16); γ_9 – число КЛОА, выполненных функцией Reduce в процессе агрегирования атрибутов атр1, атр2, ..., атрq, рассчитывают по формуле:

$$\gamma_9 = Q_J(3q+1). \qquad (29)$$

Ниже приведено выражение для оценки объема записей, получаемых после агрегирования значений атрибутов и сохраняемых MR в базе данных NoSQL:

$$V_{AY1} = \min(V_{A1}, Y \cdot l_A), \qquad (30)$$

$$V_{A1} = l_A \cdot Q_{A1X1}, \qquad (31)$$

где V_{A1} – этот объем равен произведению длины записи agr1/(A1X, agr2, ..., agrq) на число групп $Q_{A1X1} = \min(Q_J, I_{A1X}/n)$; I_{A1X} – мощность атрибута A1X. В выражении (30) учитывается ограничение на число результирующих записей (см. параметр LIMIT в запросе (1)).

Следует отметить, что сортировку записей на 2-й фазе выполнять не обязательно, но MR это делает по умолчанию.

Фаза 3. На этой заключительной фазе нужно определить только одну функцию Reduce, которая использует выходные данные предыдущей фазы для получения Y записей, упорядоченных по значению agr1 (см. запрос (1)).

Функция Map. На каждом узле запускается функция Map (метка 2 на рис. 1), которая принимает записи «ключ/значение» agr1/(A1X, agr2, ..., agrq), сохраненные в базе данных на 2-й фазе, и просто возвращает их как значение функции.

Для расчета среднего времени чтения записей из базы данных и их сортировки/объединения можно использовать формулы (19) – (25). Только в них необходимо сделать следующие замены: V_J заменить на V_{AY1} (см. (30)), а Q_J – на V_{AY1}/l_A.

Функция Reduce. Выполняется только один экземпляр этой функции в одном узле – она читает все записи, отсортированные по значению agr1, и сохраняет первые Y записей в базе данных.

Для расчета среднего времени пересылки и сортировки можно использовать следующие формулы (по аналогии с (26) – (31)):

$$T_{R3} = T_{RNW3}(V_{AY1}) +$$
$$+\max\{\frac{nV_{AY1}}{\mu_{DR3}} + t_{PR34},\ \frac{nV_{AY1}}{\mu_{DW4}}\} + \max\{\frac{nV_{AY1}}{\mu_{DR4}} + t_{PR35},\ \frac{V_{AY}}{\mu_{DW1}}\}, \quad (32)$$

$$T_{RNW3}(V_{AY1}) =$$
$$= V_{AY1}\max\{\frac{1}{\mu_{DR2}}, \frac{1}{\mu_{PW}}, (\frac{n}{k} - 1)\frac{1}{\mu_{N1}},\ (n - \frac{n}{k})\frac{1}{\mu_{N2}}, \frac{n-1}{\mu_{PR}}, \frac{(n-1)+1}{\mu_{DW2}}\}, \quad (33)$$

$$t_{PR34} = \tau\frac{nV_{AY1}/l_A}{n/u}\gamma_4, \quad (34)$$

$$t_{PR35} = \tau \cdot (nV_{AY1}/l_A)\gamma_5, \quad (35)$$

$$V_{AY} = \min(V_A,\ l_A \cdot Y),\ V_A = l_A \cdot Q_{A1X},\ Q_{A1X} = \min(Q_J,\ I_{A1X}), \quad (36)$$

$$V_{AY1} = \min(V_{A1},\ Y \cdot l_A),\ V_{A1} = l_A \cdot Q_{A1X1},\ Q_{A1X1} = \min(Q_J,\ I_{A1X}/n). \quad (37)$$

Формулы (32) – (35) учитывают, что записи, полученные на фазе 2 «*n*» узлами, обрабатываются на 3-й фазе только одним экземпляром функции Reduce. Выражение (36) учитывает, что на фазе 3 в базе данных NoSQL сохраняются не более Y записей, упорядоченных по значению arg1 (см. запрос (1)). Формула (37) учитывает, что среднее число записей, поступивших из каждого узла, не превышает величины I_{A1X}/n, где I_{A1X} мощность атрибута A1X (см. также (30)).

Среднее время выполнения запроса (1) в базе данных NoSQL по технологии MapReduce вычисляется следующим образом:

$$T_{NoSQL} = (T_{M1} + T_{R1}) + (T_{M2} + T_{R2}) + (T_{M3} + T_{R3}), \quad (38)$$

где слагаемые определяются выражениями (3), (12), (19), (26), (19), (32).

Заключение

1. Получены выражения времени выполнения запроса (1) по технологии MapReduce. Они учитывают особенности межмашинного обмена между узлами, механизмы хеширования и сортировки записей, а также многие другие особенности выполнения запроса в этой среде.

2. В модель введен параметр времени выполнения короткой логической операции алгоритма (КЛОА), позволяющий выполнять калибровку модели по результатам натурных экспериментов.

3. Но NoSQL не надо идеализировать. Эти базы данных не поддерживают блокировки записей, они не позволяют вести транзакции. Поэтому их нельзя использовать, например, в финансовых системах. Выполнение запроса необходимо вручную разбивать на фазы и реализовывать соответствующие процедуры Map и Reduce. Поэтому появился проект HadoopDB [15], в котором предпринята попытка объединить преимущества параллельных СУБД и технологии MapReduce. Но пока HadoopDB – это исследовательский прототип с ограниченными функциональными возможностями. Конечно, и для таких систем необходимо разрабатывать математические модели, позволяющие исследовать их поведение при больших объемах обрабатываемых данных.

4. В следующей публикации будут приведены примеры расчетов, а также

сделано сравнение времени выполнения запроса (1) в двух средах: в строчной параллельной СУБД и по технологии MapReduce.

ЛИТЕРАТУРА

1. *Кузнецов С.* MapReduce: внутри, снаружи или сбоку от параллельных СУБД? [Электронный ресурс] [http://citforum.ru/database/articles/dw_appliance_and_mr/5.shtml]. Проверено 23.03.2013.
2. *Павло Э., Паулсон Э., Александр Р. и др*. Сравнение подходов к крупномасштабному анализу данных. Пересказ Сергея Кузнецова. [Электронный ресурс] [http://citforum.ru/database/articles/mr_vs_dbms/]. Проверено 16.02.2013.
3. *Григорьев Ю.А., Плутенко А.Д.* Анализ процесса выполнения запроса на соединение таблиц в строчной параллельной СУБД // Информатика и системы управления. – 2013. – № 4. – С. 3-15.
4. *Редмон Э., Уилсон Д.Р.* Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. – М.: ДМК Пресс, 2013.
5. *Hadoop*, HDFS, MapReduce and Hive – Some salient [http://hadoop-gyan.blogspot.ru/].
6. *Hadoop MapReduce.* Основные концепции и архитектура (Платформа Hadoop. Часть 3). [Электронный ресурс] [http://www.codeinstinct.pro/2012/08/mapreduce-design.html] Проверено 23.03.2013.
7. *Миллер Р., Боксер Л.* Последовательные и параллельные алгоритмы: Общий подход. – М.: БИНОМ. Лаборатория знаний, 2006.
8. *Григорьев Ю.А., Плужников В.Л.* Оценка времени соединения таблиц в параллельной системе баз данных // Информатика и системы управления. – 2011. – № 1. – С. 3-16.
9. *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. – 2012. – Спец. выпуск №5. – С. 31-45.
10. *Григорьев Ю.А., Плутенко А.Д.* Теоретические основы анализа процессов доступа к распределенным базам данных. – Новосибирск: Наука, 2002.
11. *Yahoo! Hadoop Tutorial.* [http://developer.yahoo.com/hadoop/tutorial/index.html].
12. *Lewis J.* Cost-Based Oracle Fundamentals. – Apress, 2006.
13. *Graefe G.* Query Evaluation Techniques for Large Databases // ACM Computing Surveys. – 1993. – Vol. 25, No. 2. – P.73-170.
14. *Graefe G.* Implementing Sorting in Database Systems // ACM Computing Surveys. –2006. – Vol. 38, No. 3. – P.1-37.
15. *Григорьев Ю.А., Плутенко А.Д.* Теория и практика проектирования систем на основе баз данных: Учебное пособие. – Благовещенск: Амурский гос. ун-т, 2007.
16. *Michael G.N.* Benchmarking and Stress Testing an Hadoop Cluster With TeraSort, TestDFSIO & Co [http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench/].
17. *Кузнецов С.* HadoopDB: архитектурный гибрид технологий MapReduce и СУБД для аналитических рабочих нагрузок. [Электронный ресурс] [http://citforum.ru/database/articles/hadoopdb/]. Проверено 20.06.2013.

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