УДК 004.657

© 2016 г. **Ю.А. Григорьев,** д-р техн. наук,
В. А. Пролетарская
(Московский государственный технический университет им. Н.Э. Баумана)

СРАВНЕНИЕ МЕТОДОВ ОБРАБОТКИ ЗАПРОСОВ К ХРАНИЛИЩУ ДАННЫХ ПО ТЕХНОЛОГИИ MAPREDUCE

Разработан метод обработки запросов к хранилищу данных с ранней материализацией без хеширования таблиц измерений в оперативной памяти (БХТИ). Выполнено сравнение метода БХТИ и метода MapReduce-Invisible Join (MRIJ) с хешированием таблиц измерений в ОП. По результатам оценки времени выполнения запроса Q3 из тестового набора TPC-H выявлены условия применимости каждого из этих методов. Проанализированы зависимости времени выполнения запроса от количества узлов в кластере и объема данных в таблицах хранилища.

Ключевые слова: технология MapReduce, хранилище данных, хеширование таблиц, метод MRIJ, метод БХТИ, время выполнения запроса, сравнение методов.

Введение

В настоящее время одним из самых востребованных решений по разработке систем распределенной обработки данных является использование технологии MapReduce. Она позволяет распределить большой объем данных по сотням, тысячам маломощных компьютерных узлов, обеспечить их надежное хранение (посредством репликации) и параллельную обработку данных в узлах за приемлемое время. Учитывая, что кластеры с большим числом узлов можно арендовать в «облаке», а программное обеспечение для поддержки технологии MapReduce является бесплатным (Linux, Hadoop и др.), можно говорить о возможности систем MapReduce конкурировать с базами данных при решении задач аналитической обработки очень больших массивов данных (петабайты).

Концепция MapReduce создала целый набор новых парадигм и структур для создания и обработки различных типов запросов. В рамках данной концепции существует несколько методов доступа к хранилищу данных (ХД), которые различаются количеством фаз Map-Shuffle-Reduce,форматом хранения записей, распределением данных по узлам и использованием ранней или поздней материализации результатов поиска.

Алгоритмы с поздней материализацией (ПМ) обрабатывают указатели на данные (записи), а чтение самих записей ХД выполняется на последнем этапе, т.е. перед выводом результатов [1, 2]. Это ускоряет поиск при небольших значениях селективности (см. следующий раздел).

В алгоритмах с ранней материализацией (РМ) записи читаются и обрабатываются на раннем этапе. Они применяются при обработке достаточно больших объемов данных [2]. Главное достоинство ранней материализации заключается в уменьшении числа фаз Map-Shuffle-Reduce при выполнении запроса. Можно выделить алгоритмы РМ без хеширования (сохранения) таблиц измерений в оперативной памяти (ОП) [2, 3] и с их хешированием [4, 5].

В статье выполнено сравнение разработанного нового метода с ранней материализацией без хеширования таблиц измерений в оперативной памяти (БХТИ) [3] и метода MapReduce-Invisible Join (MRIJ) с хешированием этих таблиц в ОП [4].

Преимущества и недостатки методов доступа к хранилищу данных по технологии MapReduce

Методы с поздней (отложенной) материализацией позволяют уменьшить время выполнения запросов к ХД при небольших объемах обрабатываемых данных, но они имеют ряд недостатков:

с увеличением числа узлов N время выполнения запроса сначала убывает, а затем возрастает почти линейно от N в результате тиражирования фильтра Блума по всем узлам кластера; возрастают требования к объему ОП каждого узла, где выполняется объединение фильтров Блума, поступивших со всех узлов [1];

в работе [2] поздняя материализация рассматривается при случайном доступе к записям ХД с использованием индекса; было установлено, что этот вид материализации лучше ранней материализации при небольших значениях селективности (меньше $6 \cdot 10^{-5}$). При этом отношение времени случайного чтения к времени последовательного сканирования (на 1 байт) составляет $\tau 1 = 1{,}5 \cdot 10^4$, можно сделать вывод, что доступ к отдельным записям – очень затратная операция.

В работах [4, 5] рассматриваются методы ранней материализации записей хранилища данных с хешированием таблиц измерений в оперативной памяти. Эти методы эффективны, если таблицы измерений невелики и результаты их фильтрации могут быть размещены в оперативной памяти узла во время выполнения запроса. Но, например, при увеличении параметра наполнения SF до 1000 в тесте TPC-H любая из 5 таблиц измерений, размеры которых зависят от SF, не помещается в буфер ОП объемом 2 Гб (после декомпрессии). В [5] предлагается выполнить денормализацию таблицы фактов для больших таблиц измерений. Но в этом случае существенно возрастает объем ХД (в 5.1 раза для теста TPC-H) и увеличивается время загрузки хранилища. Как будет показано в следующих разделах, методы с хешированием таблиц измерений в ОП перестают работать, если хотя бы одна таблица измерения не может быть сохранена в ОП узла.

В работе [2] предлагается метод ранней материализации без хеширования таблиц измерений в ОП. Запрос к ХД выполняется следующим образом.

1. Создаются дополнительные PF-группы таблицы фактов, упорядоченные по внешним ключам (в каждую группу должен входить первичный ключ таблицы фактов).

2. Соединение выполняется в одном задании:

1) на фазе Map параллельно реализуются соединения R0><R1, … R0><Rn (n типов map) методом merge-join по первичному ключу каждого измерения, R0 – таблица фактов; Ri – таблица i-го измерения; для этого программа map а) читает партицию (split) таблицы фактов (или PF-группы), б) определяет и читает блоки таблицы измерений с диапазоном первичного ключа, который соответствует split таблицы фактов, в) последовательно читает и соединяет записи методом merge-join; материализация записи выполняется следующим образом: по значению позиции в таблице устанавливаются указатели в блоках столбцов, выполняется их сканирование и конкатенация значений;

2) на фазе Reduce записи группируются и соединяются по первичному ключу таблицы фактов.

У этого метода имеются недостатки.

1. PF-группы можно создавать при загрузке хранилища или динамически. Создание дополнительных PF-групп, распределение их по узлам – очень ресурсоемкие операции (необходима сортировка всех записей группы по внешнему ключу).

2. Новые записи надо добавлять и в основную таблицу, и в PF-группу (в отсортированном виде).

3. Блоки таблиц измерений с требуемым диапазоном первичного ключа могут храниться в узлах, отличных от того, где выполняется программа map; это приводит к росту сетевого трафика.

В работе [3] разработана модель нового метода ранней материализации записей без хеширования таблиц измерений в оперативной памяти (метод БХТИ), который не имеет указанных недостатков. В следующем разделе кратко рассмотрен алгоритм метода БХТИ.

Метод ранней материализации без хеширования таблиц измерений в оперативной памяти (БХТИ)

Ниже приведено описание метода БХТИ. Таблицы измерений и таблица фактов хранятся в формате RCFile [6]. Записи записываются в следующем формате: (ключ)(значение). Ниже символы {…} обозначают список.

Задание 1 (Job 1).

ФазаMap

1. Чтение столбцов из таблицы фактов, указанных в запросе. Проверяется подусловие поиска для значений фактов mj записи. Если запись удовлетворяет всем проверкам, то в выходной поток помещаются следующие записи:

(i vfki 1) (0 позиция X), (1)

где i – номер измерения (i = 1, 2, …, n); vfki – значение внешнего ключа i-го измерения таблицы фактов; 1 – вводится для упорядочивания значений в группе после группирования на фазе Reduce; 0 – признак таблицы фактов; «позиция» – номер

позиции в таблице фактов для i = 1 X = $\{vmj\}_j$ – множество значений фактов, указанных за SELECT, для i > 1 X = 0 – признак пустоты.

2. Чтение таблиц измерений, участвующих в запросе. Проверяется подусловие поиска. В выходной поток помещаются следующие записи:

(i vdi.0 0) (1 i Y), (2)

где i – номер измерения (i = 1, 2, …, n); vdi.0 – значение первичного ключа таблицы i-го измерения; 0 (третье поле ключа) – вводится для упорядочивания значений после группирования на фазе Reduce; 1 – признак таблицы измерения; Y = $\{vdi.j\}_j$ – множество значений таблицы i-го измерения, указанных за SELECT.

Фаза Shuffle

Распределение записей выходного потока по узлам выполняется по первой паре атрибутов ключа (i v) записей (1) и (2). Сортировка на фазе Reduce выполняется по ключу (i v j). Группирование записей входного потока (перед передачей их на вход функции Reduce) выполняется по паре атрибутов ключа (i v). Формат записи входного потока для функции Reduce имеет следующий вид:

(i v 0) ((1 i Y) {(0 позиция X)}). (3)

Примеры записей:

A B (т.е. список)

(1 v 0) ((1 1 Y) {(0 позиция X)}),

(2 v 0) ((1 2 Y) {(0 позиция X)}).

ФазаReduce

1. Если в записи отсутствует пара (A B), то эта запись исключается из рассмотрения (т.е. таблица измерения и таблица фактов не могут быть соединены по соответствующему значению ключа). Например, должны быть исключены следующие записи:

(1 v 0) (1 1 Y) – отсутствует список B (т.е. {(0 позиция X)}) – в таблице фактов нет такого значения внешнего ключа по измерению i = 1;

(2 v 0) ({(0 позиция X)}) – отсутствует поле A (т.е. (1 2 Y)) – в отфильтрованном измерении i = 2 нет соответствующего значения ключа.

2. Записи (3) помещаются в выходной поток, и затем MapReduce сохраняет их в файле HDFS, например, Ф1.

Таким образом, на фазе Reduce реализуется соединение столбцов с ключами таблиц измерений и фактов (по каждому внешнему ключу измерения таблицы фактов строится своеобразная маска).

Задание 2 (Job 2)

ФазаMap

1. Чтение записи из файла Ф1, сформированного функцией Reduce Задания 1 (см. (3)):

A B (т.е. список)

(i v 0) ((1 i Y) {(0 позицияX)}), (4)

где i – номер измерения (i = 1, 2,…, n); (1 i Y) – значения атрибутов Y = $\{vdi.j\}_j$ (то что за SELECT) в таблице i-го измерения в строке со значением первичного ключа, равным «v»; {(0 позиция X)} – список позиций записей таблицы фактов, которые имеют такое же значение внешнего ключа i-го измерения, т.е. «v»; X =

{vmj} для i = 1 и X = 0 для i > 1.

2. Сканирование списка B записи (4), и для каждого элемента этого списка в выходной поток помещаются новые записи:

(позиция i) ((0 i X) (1 i Y)), (5)

где «позиция» – это позиция из списка B.

В записях (5) для каждой позиции из списка B повторяется поле A.

Фаза Shuffle

Распределение записей (5) выходного потока по узлам выполняется по первому атрибуту ключа (позиция). Сортировка на фазе Reduce выполняется по ключу (позиция i). Группирование записей входного потока (перед передачей их на вход функции Reduce) выполняется по первому атрибуту ключа (позиция).

Формат записи входного потока для Reduce:

(позиция 1)({(0 i X) (1 i Y)}$_i$), (6)

«позиция» – позиция в таблице фактов, X = {vmj} для i = 1 и X = 0 для i > 1; (1 i Y) – это тройка, состоящая из полей; 1 – признак таблицы измерения; i – номер измерения (i = 1, 2, …, n), Y = {vdi.j}.

ФазаReduce

1. Запись (6) исключается из рассмотрения, если множество {i} в списке не совпадает с множеством (1, 2, …, n). Это означает, что кортеж с номером «позиция» исходной таблицы фактов содержит в каком-либо измерении недопустимое значение ключа (и оно было отфильтровано функцией Reduce Задания 1).

2. В выходной поток помещается результирующая запись

(позиция)({(vmj 0 j)} {(vd1.j 1 j)} {(vd2.j 2 j)} …), (7)

где vmj – значение факта; 0 – признак таблицы фактов; j – номер факта в списке (выбирается из X для i = 1 – см. (6)); vdi.j – значение j-го столбца таблицы i-го измерения (выбирается из Y).

Затем MapReduce сохраняет записи в файле Ф2.Результирующая таблица сформирована. Далее при необходимости можно выполнить группирование и сортировку записей.

Задание 3 (Job 3)

Выполняется, если требуется выполнить группирование и сортировку результирующих записей.

ФазаMap

1. Чтение записи из файла Ф2, сформированного функцией Reduce Задания 2 (см. (7)).

2. В выходной поток помещается запись

({(vdi.j i j)} позиция) ({(vmj j позиция)}), (8)

где {(vdi.j i j)} – подмножество значений в списке значений измерений в (7), по которым выполняется группировка/сортировка; {(vmj j позиция)} – подмножество значений фактов, для которых выполняются функции агрегирования; j – номер факта; «позиция» – позиция в таблице фактов.

Группирование записей (8) входного потока (перед передачей их на вход функции Reduce) выполняется по списку {(vdi.j i j)}, «позиция» в ключе используется в процессе сортировки исходных записей. Формат записи входного потока

для функции Reduce имеет вид:

({(vdi.j i j)} 0)({{(vmj j позиция)}$_j$}$_{позиция}$) (9)

ФазаReduce

1. Выполняет агрегирование значений фактов и в выходной поток помещаются записи

({(vdi.j i j)}) ({(agr_vmj j)}), (10)

где agr_vmj – агрегат j-го факта (SUM, AVG, MIN, MAX и др.).

MapReduce сохраняет записи (10) в файловой системе HDFS, например, под именем Ф3.

Рассмотренный выше новый метод доступа к ХД имеет следующие преимущества:

1) отсутствует дублирование таблиц измерений в узлах, и поэтому нет необходимости передачи их фрагментов по сети;

2) нет необходимости хешировать таблицы измерений в оперативной памяти;

3) таблицы измерений и фактов могут быть равномерно распределены по узлам, не надо менять политику распределения блоков, принятую в MapReduce по умолчанию;

4) используется унифицированный метод хранения таблиц (RCFile), позволяющий выполнять декомпрессию только тех столбцов, которые используются в запросе.

Математическая модель процессов обработки запросов к хранилищу данных для рассмотренного выше метода приведена в работе [3].

Метод ранней материализации с хешированием таблиц измерений в оперативной памяти (MRIJ)

В [4] рассматривается метод доступа MapReduce-InvisibleJoin (MRIJ) к хранилищу данных с большими таблицами измерений. Таблицы измерений достаточно велики и их фрагменты, удовлетворяющие условию запроса,могут быть сохранены в ОП узла, но по отдельности.

Рассмотрим случай, когда в одном узле хранится одна большая таблица измерений. В этом же узле должна храниться колонка соответствующего внешнего ключа таблицы фактов [4]. Это непростая задача, и она может быть решена с использованием надстройки CoHadoop [7] (рис. 1).

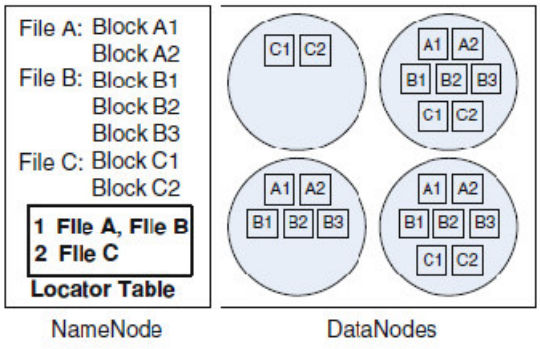

Рис. 1. Совместное хранение блоков файлов A и B в одном узле, а также их реплик в CoHadoop.

Ниже приведено описание метода MRIJ с одной большой таблицей измерения [4].

Задание 1 (Job 1).

Фаза Map

1. В узле читаются записи таблицы измерения; выполняется фильтрация записей, удовлетворяющих условию запроса; записи хешируются в ОП (если места в ОП недостаточно, то они сохраняются на диске). Читается колонка соответствующего внешнего ключа таблицы фактов и проверяется наличие значений этого ключа в хеше (если требуемой записи таблицы измерения не обнаружено, то сканируются записи, сохраненные на диске); при успешном сравнении в выходной поток выводятся записи:

(позиция)(value), (11)

позиция – номер строки таблицы фактов, value – список требуемых значений столбцов строки таблицы измерения (выбираются из хеша).

2. В других узлах параллельно читаются значения из колонок фактов, выполняется проверка условия запроса, в выходной поток помещаются записи:

(позиция)(vmj), (12)

где vmj – значение j-го факта.

Фаза Reduce

1. Если в записи, полученной функцией Reduce (после группирования по позиции), число элементов в области значения равно 1+k (1, k – число требуемых измерений и столбцов фактов) и она удовлетворяет условию запроса (проверяются заданные отношения между столбцами), то эта запись помещается в выходной поток как строка результирующей таблицы:

(позиция)(value,{vmj}), (13)

где value – список значений требуемых столбцов таблицы измерения.

Далее при необходимости можно выполнить группирование и сортировку записей по аналогии с Заданием 3 (см. предыдущий раздел).

Можно отметить следующие недостатки рассмотренного метода:

1) вместе с таблицей измерения в узле должна храниться соответствующая колонка внешнего ключа таблицы фактов; для этого необходимо менять политику распределения блоков файловой системы по узлам системы, принятую в MapReduce по умолчанию;

2) если записи таблицы измерения, удовлетворяющие запросу, не могут быть сохранены в ОП узла, то необходимо сканировать сохраненные на диске записи.

В соответствии с рассмотренным алгоритмом была разработана математическая модель, позволяющая оценивать время выполнения запроса к ХД с большой таблицей измерения (в статье она не приводится).

Сравнение методов БХТИ и MRIJ

В качестве примера для сравнения был взят запрос Q3 к хранилищу данных из теста TPC-H [8]. Схема «снежинка» хранилища была денормализована в схему «звезда» (рис. 2). Это позволило избежать соединения таблиц измерений при вы-

полнении запроса. В этом случае запрос Q3 к ХД имеет следующий вид:

```
SELECT l_orderkey, sum(l_extendedprice*(1-l_discount)) as revenue, o_orderdate,
o_shippriority
FROM orders o, lineitem l
    WHERE o_mktsegment = '[SEGMENT]'
    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.
```

Здесь SF – это параметр теста TPC-H, влияющий на число записей таблиц измерений и фактов, загружаемых в хранилище данных (см. рис. 2).

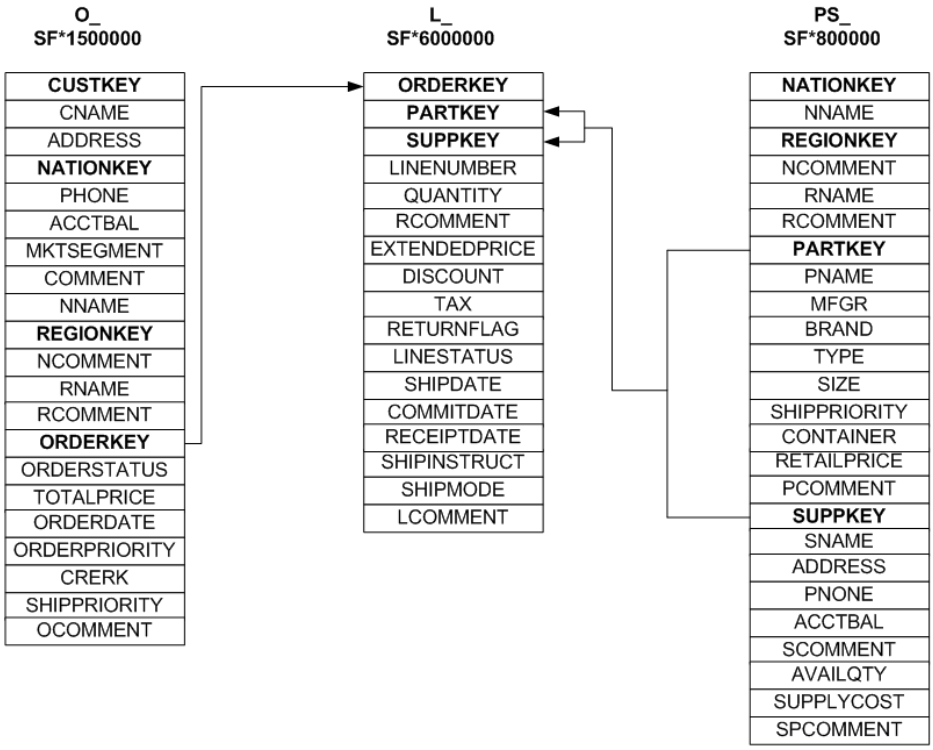

Рис. 2. Денормализованная схема хранилища данных теста TPC-H.

При исследовании метода БХТИ учитывалось, что таблицы измерений и фактов равномерно распределены по узлам кластера MapReduce. Для метода MRIJ таблица измерения Orders и колонка внешнего ключа ORDERKEY таблицы фактов Lineitem хранятся на одном узле, записи остальных колонок таблицы фактов равномерно распределены по остальным узлам. Данные хранятся на диске в формате RCFile, метод сжатия – Lzo.

Для модельного эксперимента были использованы параметры хранилища, запроса, узлов и сети (табл. 1).

При выполнении моделирования одновременно варьировалось количество узлов Nв сети и параметр SFтеста TPC-H (рис. 3). Время выполнения запроса Q3 с использованием метода MRIJ резко возрастает при N = SF>200. Это связано с тем, что при SF > 200 объем хешируемой таблицы измерения Orders превышает размер свободной оперативной памяти, отводимой одному экземпляру Map (512-100 = 412 Мбайт – размер «кучи» минус объем буфера под сортировку на фазе

Map), и требуемые записи читаются с диска. В то же время метод БХТИ демонстрирует хорошую масштабируемость: при увеличении числа узлов и объема ХД в 10 раз (с 100 до 1000) время возросло только в 1,4 раза (с 35 с до 49 с).

Таблица 1

Характеристика	*Значение*	*Характеристика*	*Значение*
Число записей в таблице измерения orders (O_)	SF*1 500 000	Длина записи таблицы измерения O_ после денормализации	535 байтов
Число записей в таблице фактов lineitem (L_)	SF*6 000 000	Длина записи таблицы фактов L_	112 байтов
Доля записей таблицы измерения O_ в условии запроса	0,43*(1/5)	Доля записей таблицы фактов L_ в условии запроса	1-0,43=0,57
Длина атрибутов таблицы измерения O_, указанных за select	3+4=7 байтов	Длина атрибутов таблицы фактов L_, указанных за select	8+4+4=16 байтов
Узел: один процессор Intel Core 2 Duo, 2,40 Ггц, ОС Red Hat Enterprise Linux 5 (версия ядра 2.6.18)	4 Гбайтов ОП, два 250-Гбайтных диска SATA-I	Производительность ввода/вывода на локальный диск	50 Мбайтов/с
Процессорное время одной КЛОА [9]	$1{,}6 \cdot 10^{-8}$ с.	Производительность ввода и вывода файловой системы HDFS	50Мбайтов/с и 24Мбайтов/с
Число задач Map на узел	2	Число задач Reduce на узел	2
Объем буфера под сортировку на фазе Map	100 Мбайтов	Максимальное число файлов в группе на фазе Map	100
Интенсивность декомпрессии в процессоре	300 Мбайт/с	Коэффициент сжатия [2]	2,3
Сеть			
Число узлов на 1 коммутатор	50	Пропускная способность порта коммутатора	1 Гбит/с
Пропускная способность коммутирующей матрицы коммутатора	128 Гбит/с	Пропускная способность соединительного кольца коммутатора	64 Гбит/с

Рис. 3. Время выполнения запроса Q3 для случая SF = N.

На рис. 4 показана зависимость времени выполнения запроса от числа узлов N при фиксированном значении параметра наполнения ХД (SF = 200 – это значение является критическим для метода MRIJ). Из рисунка видно, что для метода MRIJ время практически не уменьшается, начиная с N = 70. Для метода БХТИ время выполнения запроса уменьшается вплоть до N = 100. При изменении N от 10 до 100 время уменьшилось: для MRIJ в 1,3 раза, для БХТИ – в 8,5 раза.

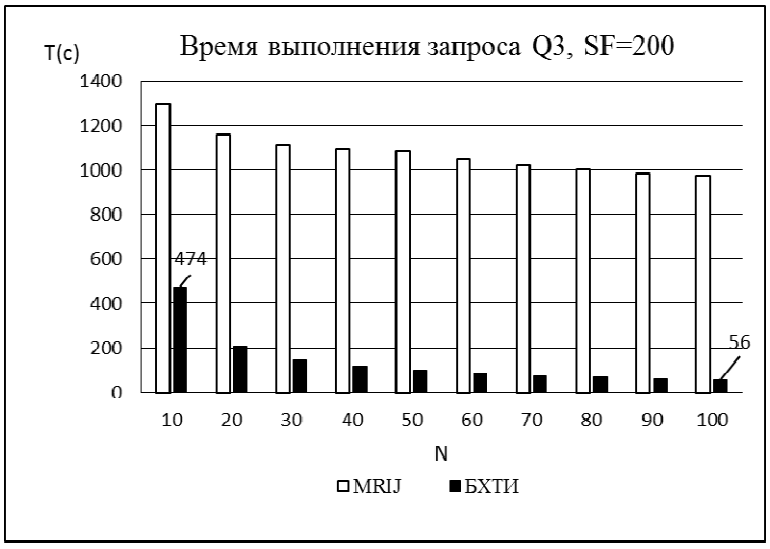

Рис. 4. Время выполнения запроса Q3 для случая SF=200.

На рис. 5 приведены графики зависимости времени выполнения запроса от значения параметра SF наполнения хранилища данных (для N = 10). Преимущество метода MRIJ наблюдается только при малых SF<1,5 (SF = 1,5 соответствует объему ХД, равному 1,5 Гбайтов). Хотя в этом случае записи таблицы измерения Orders полностью хешируются в оперативной памяти, метод БХТИ имеет преимущество при SF>1,5. Это объясняется тем, что при использовании метода MRIJ «узким местом» становится узел, где хранятся вся таблица измерения Orders и колонка внешнего ключа таблицы фактов.

Рис. 5. Зависимость времени выполнения запроса для различных значений параметра SF.

Заключение

Выявлены преимущества и недостатки существующих методов доступа к хранилищу данных по технологии MapReduce.

Разработан метод ранней материализации БХТИ, позволяющий избежать хранения записей таблиц измерений в оперативной памяти во время выполнения запроса к хранилищу данных и обеспечивающий равномерное распределение данных хранилища по узлам системы.

Выполнено сравнение метода БХТИ с методом MRIJ на примере запроса Q3 теста TPC-H.Результаты моделирования показали, что метод MRIJ перестает работать при значениях параметра наполнения хранилища SF>200. В то же время разработанный метод БХТИ демонстрирует хорошую масштабируемость: при увеличении числа узлов и объема ХД в 10 раз время возросло только в 1,4 раза. Метод MRIJ имеет преимущества при небольших объемах обрабатываемых данных (меньше 1,5 Гбайтов).

ЛИТЕРАТУРА

1. *Григорьев Ю.А., Плутенко А.Д., Плужников В.Л., Ермаков Е.Ю., Цвященко Е.В., Пролетарская В.А.* Теория и практика анализа параллельных систем баз данных. – Владивосток: Дальнаука, 2015.
2. *Lin Y., Agrawal D., Chen C., Ooi B.C.,Wu S.* Llama: leveraging columnar storage for scalable join processing in the MapReduce framework // Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. – P.961-972.
3. *Григорьев Ю.А. Пролетарская В.А.* Метод ранней материализации доступа к хранилищу данных по технологии MapReduce // Информатика и системы управления. – 2015. – № 3. – С.3-16.
4. *Zhou G., Zhu Y., Wang G.* Cache Conscious Star-Join in MapReduce Environments. Cloud-I '13 Proceedings of the 2nd International Workshop on Cloud Intelligence, August 26, 2013.
5. Songting Chen Turn Inc. Cheetah: A High Performance, Custom Data Warehouse on Top of MapReduce // Journal Proceedings of the VLDB Endowment. – 2010. – Vol. 3, Issue 1-2, P.1459-1468.
6. *Lee Rubao, Huai Yin, Shao Zheng,* etc. RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems // ICDE. – 2011. – P.1199-1208.
7. *Eltabakh M.Y., Tian Y., O¨zcan F., Gemulla R., Krettek A.a, McPherson J.* CoHadoop: Flexible Data Placement and Its Exploitation in Hadoop // Journal Proceedings of the VLDB Endowment. – 2011. – Vol. 4, Issue 9. – P.575-585.
8. Документация TPC-H. (http://www.tpc.org/tpc_documents_current_versions/pdf/tpch2.17.1.pdf).
9. *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;
Пролетарская Виктория Андреевна – vilka2000@mail.ru.