УДК 004.657

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

МЕТОД РАННЕЙ МАТЕРИАЛИЗАЦИИ ДОСТУПА К ХРАНИЛИЩУ ДАННЫХ ПО ТЕХНОЛОГИИ MAPREDUCE

Разработан новый метод доступа к хранилищу данных со схемой «звезда», реализованному в среде MapReduce. Он включает три последовательно выполняемых задания. Первое задание строит маски по измерениям таблицы фактов, второе – выполняет пересечение этих масок, а третье задание осуществляет группирование, агрегирование и получение конечного результата. Определены преимущества этого метода по сравнению с существующими подходами. Разработана модель оценки времени выполнения запроса к хранилищу. В качестве примера получена оценка времени выполнения запроса Q3 тестового набораTPC-H, подтвердившая эффективность разработанного метода.
Ключевые слова: технология MapReduce, хранилище данных, ранняя материализация, время выполнения запроса.

Введение

Существует несколько подходов к построению систем, обеспечивающих распределенную обработку данных. Одним из наиболее востребованных подходов является использование парадигмы MapReduce (MR), согласно которой фрагменты данных сохраняются на многих маломощных узлах кластера, организуется их параллельная обработка. Эта модель впервые разработана и апробирована компанией Google [1].

В статье [2] рассматриваются четыре варианта реализации соединения таблиц измерений и фактов, которые связаны по схеме «звезда». Они позволяют избежать перемещений записей таблицы фактов хранилища данных (ХД) в системе MapReduce. Но эти способы выполнения запросов к хранилищу данных имеют следующие общие недостатки:

1. Все таблицы измерений, участвующие в запросе, должны быть продублированы на всех узлах системы (варианты 1 – 3). Для этого требуются дополнительные ресурсы памяти и временные затраты.

2. В варианте 4 самая большая таблица измерения должна храниться в узле вместе с колонкой соответствующего внешнего ключа таблицы фактов. Для этого необходимо изменить политику распределения блоков файловой системы по уз-

лам системы, что может привести к нарушению равномерного распределения блоков таблиц по узлам системы.

3. Хеш-индексы измерений, участвующих в запросе, должны храниться в ОП узла. Если в качестве измерения выступает большая полноценная таблица (например, «сотрудник», «болезнь», «спутник» и т.д.) и хеш-индексы велики, то могут возникнуть проблемы с оперативной памятью, так как в узлах MapReduce используются маломощные станции.

4. Имеет место пересылка лишних записей, что приводит к увеличению объема данных, передаваемых узлам на фазе shuffle, т.е. в этих методах не удалось избежать пересылки значений измерений и фактов между узлами.

В данной статье предлагается метод ранней материализации (MPM) доступа к хранилищу данных, который лишен многих из указанных недостатков.

Алгоритм функционирования системы MapReduce

Фаза Map (рис. 1). На каждом узле M_i запускается L экземпляров функции Map (метка 2, метки на рис. 1 подчеркнуты), которые выполняются параллельно и читают из файловой системы MR записи (объекты) некоторой таблицы в виде пары <ключ, значение> (метка 1). Принятые записи преобразуются в новые записи <ключ1, значение1> и выводятся в буфер памяти (метка 3).

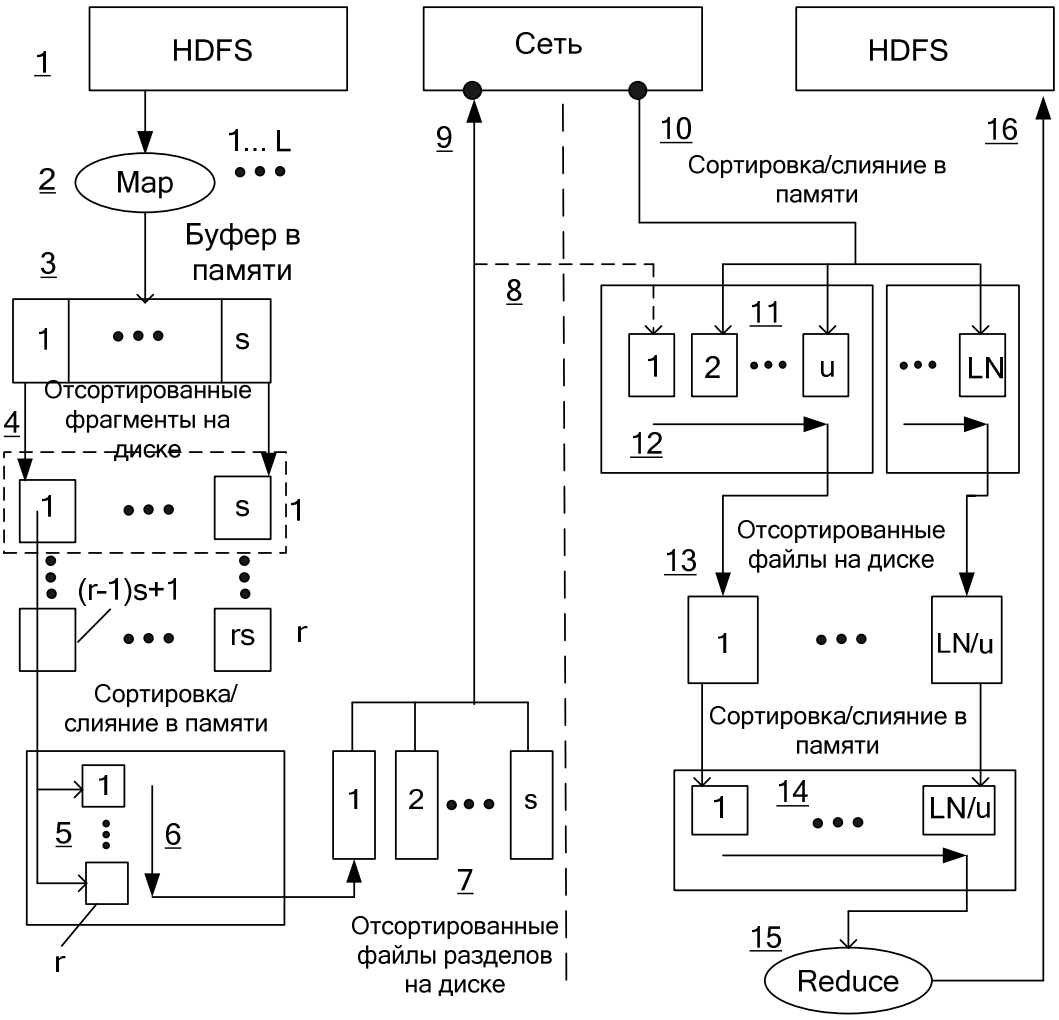

Рис. 1. Последовательность выполнения операций по технологии MapReduce.

Каждой функции Map выделяется буфер памяти объемом Q_B. Когда объем заполнения буфера превышает определенный порог, то запускается отдельная задача, которая выполняется в фоновом режиме. Она разбивает буфер на фрагменты и сортирует каждый фрагмент по ключу (in-memorysort). Число фрагментов в буфере равно числу разделов s (функций Reduce), указанному в задании (Job). Запись принадлежит *i*-му фрагменту буфера (т.е. *i*-му разделу), если h(атрибут) = i, где h – некоторая хеш-функция. После этого каждый отсортированный фрагмент буфера выводится на диск как часть файла (spilltodisk) (метка 4). Таким образом, общее число отсортированных фрагментов, которые будут сохранены на диске в процессе выполнения всех функций Map узла, будет равно r·s·L, где r – число сохраненных файлов – рассчитывается по формуле [3]:

$$r = Q/(Q_B \cdot P_T \cdot P_M/16) = Q/Q_M, \quad (1)$$

здесь Q – общее число выходных записей, сформированных одним экземпляром функции Map; Q_B – размер буфера памяти (по умолчанию 100 Мбайт); P_T – порог заполнения буфера (0.8); P_M – доля буфера, отведенного под метаданные (0.05).

Примечание:

фрагменты (j – 1)s + 1,…, js – это части *j*-го файла, j = 1,…, r (рис. 1, метка 4), т.е. на диске сохраняется r физических файлов, каждый из которых содержит s логических файлов (фрагментов);

если задействована возможность Combiner (по умолчанию она отключена), то сначала каждый фрагмент, сформированный в буфере памяти, подается на вход функции Reduce, и только потом результат выполнения функции Reduce сохраняется на диске (см. метку 4). Для некоторых задач это позволяет уменьшить объем обрабатываемых данных.

После того, как функция Map обработает все входные записи, в узле запускается специальная процедура. Она сортирует записи *i*-го раздела и объединяет их в один файл (mergeondisk). Для этого процедура выделяет в памяти r блоков – один на каждый фрагмент *i*-го раздела (метка 5). И читает записи *i*-го фрагмента из 1-го файла в 1-й блок, из 2-го файла – во 2-й блок и т.д., из r-го файла – в *r*-й блок. В каждом блоке записи уже отсортированы на предыдущем этапе. Поэтому сравниваются первые записи этих блоков по ключу «ключ1» (r сравнений). Запись с минимальным значением ключа перемещается в буфер диска (одно перемещение) (метка 6). Остальные записи соответствующего блока сдвигаются вправо (блок работает как стек, точнее, указатель в блоке смещается влево). Затем снова сравниваются первые записи r блоков по ключу и т.д. Если записи в какомлибо блоке ОП исчерпаны, то в этот блок подгружаются записи из связанного с ним файла. После обработки r фрагментов будет сформирован отсортированный по ключу «ключ1» файл *i*-го раздела(метка 7). Описанная выше процедура последовательно выполняется для всех разделов (i = 1,…, s) экземпляра Map.

Указанные выше действия выполняются параллельно для всех экземпляров Map, запущенных на данном узле. Таким образом будет сформировано L файлов для каждого раздела. Сортировка необходима для выполнения операции группирования на фазе Reduce.

Фаза Shuffle. После завершения выполнения всех функций Map система

MR дает команду, узел открывает файлы, подготовленные Map (см. метку 7 на рисунке 1), читает их и передает записи узлам, где выполняются функции Reduce (метки 9, 10). Файлы разных узлов с одним номером раздела (i = 1,…, s) будут переданы в один узел.

Фаза Reduce. Узел j принимает записи, сохраняет их в файлах (рис. 1, метка 11) и объединяет их в один отсортированный массив. Для этого исходные файлы группируются по «u» файлов в группе (по умолчанию u = 100). 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, который без промежуточного запоминания на диске передается на вход функции Reduce.

Перед тем как передать запись-группу на вход функция Reduce, система группирует входные записи по значению ключа, по которому выполняется сортировка (метка 15). Далее функция Reduce обрабатывает каждую группу и помещает результат в выходной поток. Эти данные сохраняются в файловой системе MR (метка 16).

После успешного завершения выполнения задания результаты размещаются в нескольких физических файлах, образуя один распределенный файл с именем, который указал пользователь. Обычно не требуется объединять их в один файл, потому что часто полученные файлы используются в качестве входных для запуска следующего MR-задания или в каком-либо другом распределенном приложении, которое может получать входные данные из нескольких файлов.

Модель выполнения запроса к многомерному хранилищу данных

Метод ранней материализации (МРМ) доступа к хранилищу данных (ХД) реализован в виде трех последовательно выполняемых заданий, каждое из которых в общем случае включает фазы Map, Shuffleи Reduce (см. предыдущий раздел). Описания моделей выполнения этих заданий приведены в табл. 1 – 3.

Таблица 1

Задание 1

№	Ввод данных с диска	Обработка в процессоре	Вывод данных на диск	Сеть
1	Распространение функций Map и Reduce по узлам системы (N узлов).			
				T_N^Z= 5 с.
	Фаза MAP			
2	1. Чтение данных таблиц измерений и таблицы фактов, их фильтрация; $\Gamma_k = 1 + 2A_k + o_k$, где A_k, o_k – число атрибутов и логических операций в условии поиска в таблице R_k[6]; $\Gamma_F = 1 + 2A_F + o_F$ (метка 2 на рис.1).			
	$T_R^D(V(1,\Lambda,Q)) +$ $T_R^D(V(1,\Lambda_F,Q_F,1))$.	$T^P(Q,\Gamma) +$ $T^P(Q_F,\Gamma_F,1)$.		

Продолжение табл. 1

	2. Сортировка ключей в буфере и их последующая запись на диск (метки 3, 4). $\Gamma_{24}^{sort} = 6{,}5Q_{24} \times \log_2(Q_{24}/L/r_{24}/(s = N))$, где $Q_{24} = \sum P \cdot Q + nP_F Q_F$ – число записей, помещаемых в выходной поток; $V_{24} = V(1, l_{24}, P_F Q_F, 1) + V(P, U, Q)$ – объем данных в выходном потоке; $l_{24} = ((1+4+1)+(1+8+1))n - 1 + \sum_j l_{vmj}$ – длина записи выходного потока (ВП) для таблицы фактов; $U_i = (1+4+1)+(1+1+\sum_j l_{vdi.j})$ – длина записи ВП для таблицы измерений.			
		$T^P(1, \Gamma_{24}^{sort}, 1)$	$T_W^d(V_{24})$	
3	1. Чтение записей фрагментов и сортировка слиянием (метки 5, 6), $\Gamma_{25}^{merge} = 2{,}5(r_{24} - 1) + 4$.			
	$T_R^d(V_{24})$	$T^P(Q_{24}, \Gamma_{25}^{merge}, 1)$		
	2. Запись в локальный файл (от всех экземпляров Map, метка 7).			
			$T_W^d(V_{24}, 1)$	
	Фаза SHUFFLE			
4	Передача промежуточных файлов системы (перетасовка, метки 8, 9).			
				$T_{SH}^N(V_{24})$
	Фаза REDUCE			
5	1.Чтение записей и промежуточная сортировка методом слияния (метки 10, 11, 12), $\Gamma_{27}^{merge} = 2.5(\min(NL, u) - 1) + 4$.			
	$T_R^d(V_{24}, 1)$	$T^P(Q_{24}, \Gamma_{27}^{merge}, 1, 1)$		
	2. Запись данных на локальный диск (метка 13).			
			$T_W^d(V_{24}, 1)$	
6	1. Чтение записей и их окончательная сортировка методом слияния (метка 14), $\Gamma_{28}^{merge} = 2.5(\frac{N \cdot L}{u} - 1) + 4$.			
	$T_R^d(V_{24}, 1)$	$T^P(Q_{24}, \Gamma_{28}^{merge}, 1, 1)$		
	2. Группы передаются функции Reduce, там фильтруются и записываются на диск (метки 15, 16). Число-групп записей, помещаемых в файл HDFS (Ф1): $Q_{28} = \sum P \cdot Q$. Средняя длина записи: $L_{28} = (1+4+1) + [V(1, l_{24} - 6n, P_{F1} P_F Q_F, 1) + V(P, U - 6, Q)] / \sum P \cdot Q$, $P_{F1} = 1 - \prod_{i=1}^{n} (1 - Pr_i)$, где Pr_i – вероятность, что запись таблицы фактов удовлетворяет условию по *i*-му измерению. Объем файла Ф1: $V_{28} = Q_{28} \cdot L_{28}$.			
			$T_W^D(V_{28}, 1)$	

На фазе Reduce задания 1 реализуется соединение столбцов с ключами таблиц измерений и фактов (по каждому внешнему ключу измерения таблицы фактов строится своеобразная маска, см. [5], пункт 11.2.2, задание 2).

Таблица 2

Задание 2

№	Ввод данных с диска	Обработка в процессоре	Вывод данных на диск	Сеть
1	Распространение функций Map и Reduce по узлам системы (N узлов).			
				T_N^Z
	Фаза MAP			
2	1. Читает файл Ф1 из HDFS и помещает в буфер «позиции B» (метка 2 на рис. 1).			
	$T_R^D(V_{28})$			
	2. Сортировка ключей в буфере и их последующая запись на диск (метки 3, 4). $\Gamma_{32}^{sort} = 6{,}5Q_{32} \times \log_2(Q_{32}/L/r_{32}/(s=N))$; $Q_{32} = [V_{28} - 6Q_{28} - V(P,U-6,Q)]/[(l_{24} - 6n)/n] = nP_{F1}P_FQ_F$; $V_{32} = (8+1)\cdot Q_{32} + V(1, U-6, P_{F1}P_FQ_F) + V(1, l_{24} - 6n - 8n + n, P_{F1}P_FQ_F, 1)$.			
		$T^P(1, \Gamma_{32}^{sort}, 1)$	$T_W^d(V_{32})$	
3	1. Чтение записей фрагментов и сортировка слиянием (метки 5, 6), $\Gamma_{33}^{merge} = 2{,}5(r_{32} - 1) + 4$.			
	$T_R^d(V_{32})$	$T^P(Q_{32}, \Gamma_{33}^{merge}, 1)$		
	2. Запись в локальный файл (от всех экземпляров Map, метка 7).			
			$T_W^d(V_{32}, 1)$	
	Фаза SHUFFLE			
4	Передача промежуточных файлов системы (перетасовка, метки 8, 9).			
				$T_{SH}^N(V_{32})$
	Фаза REDUCE			
5	1.Чтение записей и промежуточная сортировка методом слияния (метки, 10, 11, 12), $\Gamma_{35}^{merge} = 2.5(\min(NL, u) - 1) + 4$.			
	$T_R^d(V_{32}, 1)$	$T^P(Q_{32}, \Gamma_{35}^{merge}, 1, 1)$		
	2. Запись данных на локальный диск (метка 13).			
			$T_W^d(V_{32}, 1)$	
6	1. Чтение записей и их окончательная сортировка методом слияния (метка 14), $\Gamma_{36}^{merge} = 2.5(\frac{N \cdot L}{u} - 1) + 4$.			
	$T_R^d(V_{32}, 1)$	$T^P(Q_{32}, \Gamma_{36}^{merge}, 1, 1)$		
	2. Группы передаются функции Reduce, там фильтруются и записываются на диск (метки 15, 16). Число групп-записей, прошедших фильтрацию: $Q_{36} = P_FQ_F\prod_{i=1}^{n} Pr_i$. Средняя длина группы (результирующей записи): $L_{36} = 8 + \sum_j (2 + l_{vmj}) + \sum_i \sum_j (2 + l_{vdi.j})$. Объем файла Ф2, помещаемого в HDFS: $V_{36} = L_{36}Q_{36}$.			
			$T_W^D(V_{36}, 1)$	

На фазе Reduce задания 2 определяются номера кортежей таблицы фактов, удовлетворяющих условию запроса, и соответствующие значения столбцов в таб-

лицах измерений. Это результат пересечения масок, полученных после выполнения задания 1 (см. [5], пункт 11.2.2, задание 3).

Результирующая таблица сформирована. Далее при необходимости можно выполнить группирование и сортировку записей (задание 3 в табл. 3).

Таблица 3

Задание 3

№	Ввод данных с диска	Обработка в процессоре	Вывод данных на диск	Сеть
1	Распространение функций Map и Reduce по узлам системы (N узлов).			
				T_N^Z
	Фаза MAP			
2	1. Читает файл Ф2 из HDFS (метка 2 на рис. 1).			
	$T_R^D(V_{36})$			
	2. Сортировка ключей в буфере и их последующая запись на диск (метки 3, 4). $\Gamma_{42}^{sort} = 6{,}5Q_{42} \times \log_2(Q_{42}/L/r_{42}/(s=N))$; $Q_{42} = Q_{36}$; где $L_{42} = \sum_i \sum_j (2 + l_{vdi.j}) + 8 + \sum_j (1 + 8 + l_{vmj})$ – сумма берется по тем столбцам таблиц измерений и фактов, которые участвуют в группировке и агрегировании; $V_{42} = L_{42}Q_{42}$.			
		$T^P(1, \Gamma_{42}^{sort}, 1)$	$T_W^d(V_{42})$	
3	1. Чтение записей фрагментов и сортировка слиянием (метки 5, 6), $\Gamma_{43}^{merge} = 2{,}5(r_{42} - 1) + 4.$			
	$T_R^d(V_{42})$	$T^P(Q_{42}, \Gamma_{43}^{merge}, 1)$		
	2. Запись в локальный файл (от всех экземпляров Map, метка 7)			
			$T_W^d(V_{42}, 1)$	
	Фаза SHUFFLE			
4	Передача промежуточных файлов системы (перетасовка, метки 8, 9).			
				$T_{SH}^N(V_{42})$
	Фаза REDUCE			
5	1.Чтение записей и промежуточная сортировка методом слияния (метки 10, 11, 12), $\Gamma_{45}^{merge} = 2.5(\min(NL, u) - 1) + 4.$			
	$T_R^d(V_{42}, 1)$	$T^P(Q_{42}, \Gamma_{45}^{merge}, 1, 1)$		
	2. Запись данных на локальный диск (метка 13).			
			$T_W^d(V_{42}, 1)$	
6	1. Чтение записей и их окончательная сортировка методом слияния, группирование (метка 14), $\Gamma_{46}^{merge} = 2.5(\frac{N \cdot L}{u} - 1) + 4,$ где $\Gamma_{46}^{group} = 1 + 1 + 1 + 1$ – перемещение указателя, сравнение значения составного ключа, перемещение указателя в группе, перезапись составного значения записи в группу.			
	$T_R^d(V_{42}, 1)$	$T^P(Q_{42}, \Gamma_{46}^{merge}, 1, 1) +$ $T^P(Q_{42}, \Gamma_{46}^{group}, 1, 1)$		

Продолжение табл. 3

	2. Группы передаются функции Reduce, и там выполняется агрегирование (метки 15, 16). Число групп: $Q_{46} = \prod_i I_i$, где I_i – мощность атрибута таблицы измерения, участвовавшего в группировании, $I_i \le P_i Q_i$. Средняя длина результирующей записи: $L_{46} = \sum_i \sum_j (2 + l_{vdi.j}) + \sum_j (1 + l_{vmj})$. Число КЛОА, выполняемых при агрегировании, на одну группу: $\Gamma_{46}^{agr} = (1+1+1)K + 1$ – для каждого столбца таблицы фактов, участвующего в агрегировании (K), выполнить перемещение указателя в группе, перемещение указателя в выходном буфере, агрегирование (1+1+1); переместить результирующую запись в выходной поток (1). Объем файла Ф3, помещаемого в HDFS: $V_{46} = L_{46} Q_{46}$.			
		$T^P(Q_{46}, \Gamma_{46}^{agr}, 1, 1)$	$T_W^D(V_{46}, 1)$	

Чтение требуемых значений таблиц измерений и фактов выполняется уже в задании 1, поэтому метод получил название «метод ранней материализации». Предполагается, что таблицы измерений и фактов хранятся в формате RCFile [4]. В монографии [5] рассмотрен алгоритм выполнения запроса к ХД с использованием метода с отложенной материализацией (МОМ). Табл. 1 – 3 раскрывают особенности метода ранней материализации, где использовались следующие функции:

1) время shuffle

$$T_{SH}^N(V[,x_1][,x_2][,x_3][,x_4]) = V \max\{\frac{1}{\mu_{DR}}, \frac{x_1}{\mu_{PW}}, x_2(K-1)K\frac{1}{\mu_{N1}}, x_2(N-K)N\frac{1}{\mu_{N2}}, \frac{x_3}{\mu_{PR}}, \frac{x_4}{\mu_{DW}}\}, \quad (2)$$

где по умолчанию $x_1 = 1 - 1/N$, $x_2 = 1/N$, $x_3 = 1 - 1/N$, $x_4 = 1$; N – число узлов; K – число узлов, подключенных к одному коммутатору;V – объем файла, читаемого с диска узла;μ_{DR}, μ_{PW}, μ_{N1}, μ_{N2}, μ_{PR}, μ_{DW} (байт/с) – это пропускные способности диска (чтение), порта коммутатора (выход), коммутирующей матрицы коммутатора, соединительного кольца, порта коммутатора (вход), диска (запись)[6];

2) вычисление объема данных

$$V(P, \Lambda, Q[,n]) = \sum_{k=1}^{n} P_k \Lambda_k Q_k, \quad (3)$$

где P, Λ, Q – векторы длиной n; P – вероятности; Λ – длины кортежей; Q – число записейв узле; [n] – необязательный параметр (если он отсутствует, то выбирается значение, равное числу измерений, участвующих в запросе);

3) время ввода (R) или вывода (W) на диск

$$T_{R\text{ или }W}^{d\text{ или }D}(V[,L][,\mu]) = \frac{1}{L} V \frac{1}{\mu_{R\text{ или }W}^{d\text{ или }D}}, \quad (4)$$

где L – число экземпляров Map или Reduce, запускаемых на одном узле; μ – ин-

тенсивность ввода-вывода; d или D обозначает ввод/вывод на локальный диск (d) или в файловую систему HDFS (D);

4) процессорное время

$$T^P(Q,\Gamma[,n][,L][,\tau]) = \frac{\tau}{L}\sum_{k=1}^{n} Q_k \Gamma_k, \quad (5)$$

где L – число экземпляров Map или Reduce, выполняемых в одном узле; Q – вектор числа кортежей в таблице входного узла; Γ – вектор количества коротких логических операций алгоритма(КЛОА) на один кортеж в таблицах; τ – время выполнения одной КЛОА.

$P \cdot Q = \{P_k Q_k\}$, где P – вектор вероятностей; Q – вектор числа записей.

$$\sum^{[n]} P \cdot Q = \sum_{k=1}^{n} P_k Q_k . \quad (6)$$

Чтобы получить время выполнения запроса, необходимо в таблицах просуммировать выражения, начинающиеся с символа T (время). Следует отметить одну особенность. Если в строке таблицы встречаются подстроки, разделенные пунктирной линией, то берется сумма по каждой подстроке, а значение строки таблицы – это max из полученных сумм. Например, строка 2 табл. 1 (задание 1) состоит из двух подстрок 2.1 и 2.2, разделенных пунктирной линией. Поэтому значение строки 2 будет равно

$$\max\{T_R^D(V(1,\Lambda,Q)) + T_R^D(V(1,\Lambda_F,Q_F,1)) + T^P(Q,\Gamma) + T^P(Q_F,\Gamma_F,1),$$
$$T^P(1,\Gamma_{24}^{sort},1) + T_W^d(V_{24})\}.$$

Пример расчета

В качестве примера анализа был взят запрос Q3 к хранилищу данных из теста TPC-H [7]. Схема «снежинка» хранилища была денормализована в схему «звезда» (рис. 2).

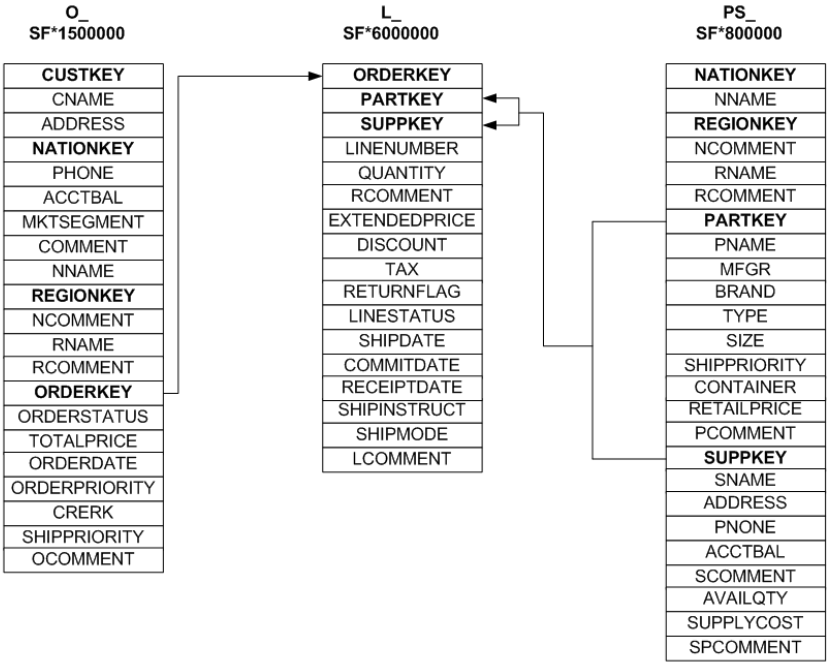

Рис. 2. Денормализованная схема хранилища данных теста TPC-H.

Это позволило избежать соединения таблиц измерений при выполнении запроса. Запрос Q3к денормализованному хранилищу имеет следующий вид:

```
SELECTl_orderkey, sum(l_extendedprice*(1 – l_discount)) asrevenue, o_orderdate,
o_shippriority
FROM orders o, lineitem l
WHEREo_mktsegment = '[SEGMENT]'ando_orderdate< date '[DATE]'
andl_shipdate> date '[DATE]'
GROUP BYl_orderkey, o_orderdate, o_shippriority
ORDER BY revenue desc, o_orderdate.
```

Для модельного эксперимента были использованы параметры хранилища, запроса, узлов и сети, представленные в табл. 4.

Таблица 4

Характеристика	Значение	Характеристика	Значение
Число записей в таблице измерения orders (O_) ($N \cdot Q$)	SF*1 500 000	Длина записи таблицы измерения O_ после денормализации (Λ)	535 байтов
Число записей в таблице фактов lineitem(L_) ($N \cdot Q_F$)	SF*6000 000	Длина записи таблицы фактов L_(Λ_F)	112 байтов
Доля записей таблицы измерения O_ в условии запроса (P_1)	0,54*(1/5)	Доля записей таблицы фактов L_ в условии запроса (P_F)	1-0,54=0,46
Длина атрибутов таблицы измерения O_, указанных за select ($\sum l_{vdi,j}$)	3 + 4 = 7 байтов	Длина атрибутов таблицы фактов L_, указанных за select ($\sum l_{vmj}$)	4+4+4=12 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 (μ_R^d, μ_W^d, μ_{DR},μ_{DW})	50 Мбайтов/с
Процессорное время одной КЛОА [8]	$1{,}6 \cdot 10^{-8}$ с	Производительность ввода/вывода файловой системы HDFS (μ_R^D, μ_W^D)	35 Мбайтов/с, 8 Мбайтов/с
Число задач Марна узел (L)	2	Число задач Reducена узел	1
Объем буфера под сортировку на фазе Map (Q_B)	100 Мбайтов	Максимальное число файлов в группе на фазе Reduce (u)	100
Сеть	варианты 1, 2		варианты 1, 2
Число узлов на 1 коммутатор (K)	12, 50	Пропускная способность порта коммутатора (μ_{PW}, μ_{PR})	1 Гбит/с
Пропускная способность коммутирующей матрицы коммутатора (μ_{N1})	24 Гбит/с, 128 Гбит/с	Пропускная способность соединительного кольца-коммутатора (μ_{N2})	8 Гбит/с, 64 Гбит/с

При выполнении модельных экспериментов вместе варьировались число узлов N и параметр SF теста TPC-H (рис. 3). Параметры сети соответствовали варианту 1 (табл. 4). Время выполнения заданий 2 и 3 практически совпадает.

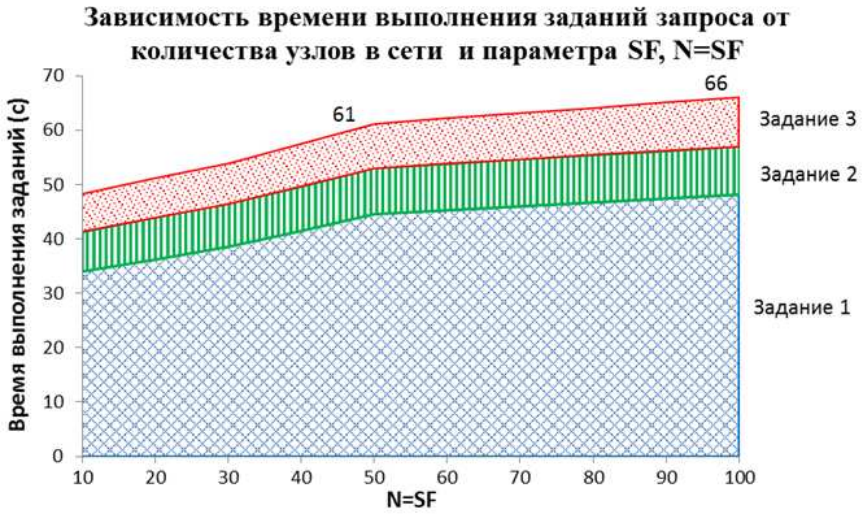

Рис. 3. Зависимость времени выполнения запроса от параметра SF и числа узлов N.

Увеличение времени выполнения запроса до точки SF = N = 50 связано с ростом времени промежуточной сортировки записей методом слиянияв задании 1 (табл.1, строка 5). Дальнейший рост времени определяется перетасовкой записей, выполняемых на фазеshuffle задания 1 (табл.1, строка 4). Система MapReduce демонстрирует хорошую масштабируемость при увеличении SF = Nc 50 до 100 время увеличилось только с 61 с до 66 с (см. рис. 3). При SF = N = 1000 время выполнения запроса составило 155с.

На рис. 4 показаны зависимости времени выполнения запроса от числа узлов в системе при различных достаточно больших значениях параметра наполнения SF хранилища данных. Здесь параметры сети соответствовали варианту 2 (см. табл. 4). Основная доля времени приходится на чтение таблиц измерений и фактов с диска в узлах (примерно 35% для SF = 3000).

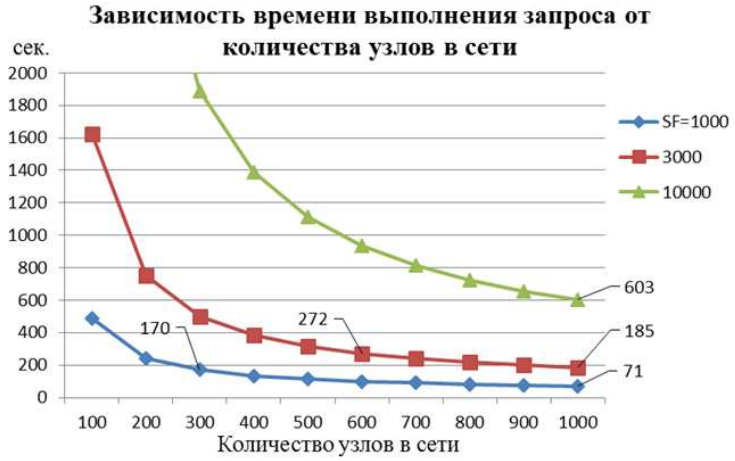

Рис. 4. Зависимость времени выполнения запроса от числа узлов N.

Сравним по критерию T·C («время·цена») рассмотренную конфигурацию MapReduce с конфигурацией CiscoUCSC460 M4 Server, которая является вторым лидером по результатам тестирования TPC-Hc параметром наполнения SF = 1000 [9], здесьT – время выполнения запроса, C – стоимость конфигурации. В тесте TPC-H параллельно выполнялись 7 потоков, в каждом потоке последовательно выполнялись 22 запроса. Каждый поток выполнялся примерно за одно и то же время, что свидетельствует о невысокой загрузке системы. Среднее время выпол-

нения запроса Q3 равно T_1 = 46с. Стоимость конфигурации составила C_1 = $571 тыс., $T_1 \cdot C_1$=26266. Для конфигурации MapReduce среднее время выполнения запроса для SF = 1000 и N = 300 равно T_2 = 170 с (см. рис. 4). В MapReduce используются маломощные станции, стоимость системного блока которых колеблется в районе $0,6 тыс. Поэтому стоимость конфигурации примерно равна C_2 = 0,6·300 = $180 тыс. (ПО здесь бесплатное), $T_2 \cdot C_2$ = 30600, т.е. конфигурация MapReduce сопоставима с конфигурацией Cisco по критерию T·C.

Заключение

Разработан новый метод (MPM) доступа к хранилищу данных по технологии MapReduce, который реализован в виде трех последовательно выполняемых заданий. Этот метод имеет следующие преимущества: здесь нет передачи (дублирования) таблиц измерений по сети; отсутствует кеширование таблиц измерений; таблицы измерений и фактов более равномерно распределены по узлам, не надо менять политику распределения блоков, принятую в MRHadoop по умолчанию; используется унифицированный метод хранения таблиц (RCFile); по сравнению с методом с отложенной материализацией [5] сократилось число заданий (с 5 до 3), необходимых для реализации алгоритма.

Разработана модель выполнения заданий MPM (табл. 1 – 3). В качестве примера промоделирован запрос Q3 теста TPC-H. Результаты моделирования показали, что конфигурация MapReduce сопоставима по критерию «время·цена» со второй в списке TPC-H@SF = 1000 конфигурацией CiscoUCSC460 M4 Server.

ЛИТЕРАТУРА

1. *Dean J., Ghemawat S.* MapReduce: Simplified Data Processing on Large Clusters. OSDI'04: Sixth Symposium on Operating System Design and Implementation. – SanFrancisco, CA, 2004.
2. *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.
3. *Palla Konstantina.* A Comparative Analysis of Join Algorithms Using the Hadoop Map/Reduce Framework. Master of Science School of Informatics Univer-sity of Edinburgh. – 2009, P.1-93.
4. *Lee Rubao, Huai Yin, Shao Zheng,* etc. RCFile: A fast and space-efficient data place-ment structure in MapReduce-based warehouse systems. ICDE 2011, P.1199-1208.
5. *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.
6. *Григорьев Ю.А., Плутенко А.Д.* Оценка времени соединения таблиц в базе данных NoSQL по технологии Mapreduce // Информатика и системы управления. – 2014. – № 1. – С. 3-16.
7. Документация TPC-H//http://www.tpc.org/tpc_documents_current_versions/pdf/tpch2.17.1.pdf
8. *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.
9. Результаты выполнения теста TPC-H // http://c970058.r58.cf2.rackcdn.com/individual_results/Cisco/cisco~tpch~1000~cisco_ucs_c460_m4_server~es~2014-12-15~v02.pdf.

E-mail:

Григорьев Юрий Александрович – grigorev@bmstu.ru;
Пролетарская Виктория Андреевна – vilka2000@mail.ru.