УДК 004.657

© 2019 г. **Ю.А. Григорьев,** д-р техн. наук,
В.А. Пролетарская
(Московский государственный технический университет им. Н.Э. Баумана)

МОДЕЛЬ ПРОЦЕССОВ ВЫПОЛНЕНИЯ ЗАПРОСОВ К ХРАНИЛИЩУ ДАННЫХ НА ПЛАТФОРМЕ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ SPARK

Осуществлен анализ процессов выполнения SQL-запросов Q3, Q17 из теста TPC-H в среде Spark. По результатам анализа разработана математическая модель этих процессов с целью оценки времени выполнения запросов к хранилищу данных для метода с каскадным использованием фильтра Блума (КИФБ). По результатам натурных экспериментов выполнена калибровка параметров разработанной модели и проанализирована ее адекватность.

Ключевые слова: SQL, Apache Spark, фильтр Блума, тест TPC-H, Big Data, моделирование.

DOI: 10.22250/isu.2019.59.3-17

Введение

В работе [1] отмечалось, что в настоящее время ведутся интенсивные исследования в направлении расширения возможностей технологии Map-Reduce (MR) для обработки больших данных (Big Data). Так, наряду с Hadoop, большую известность получила новая программная платформа Spark [2, 3]. Как и Hadoop, Spark – это не просто отдельный проект, а целая экосистема основанных на нем разнообразных проектов.

В рамках концепции MapReduce/Spark существуют специальные методы доступа к хранилищу данных (ХД). Так, Brito и другие показали эффективность использования фильтров Блума на примере хранилища данных типа «звезда» [4].

Это метод SBFCJ (Spark Bloom-Filtered Cascade Join). Он выигрывает по сравнению с методами, реализованными в MapReduce/Hadoop, по времени

реакции, объему данных, пересылаемых между узлами, и объему данных, дополнительно сохраняемых на диске [4]. Но он имеет существенные недостатки:

1) этот метод можно использовать для реализации запросов только к хранилищу данных типа «звезда»;

2) в одном запросе нельзя сочетать преимущества обоих методов: уменьшение размера промежуточных таблиц при помощи фильтра Блума и дублирование небольших таблиц по узлам системы при выполнении операций соединения.

В работах [1,5 – 7] предложен метод параллельной обработки запроса к хранилищу данных с произвольной структурой (не обязательно «звезда») с *каскадным использованием фильтра Блума* (КИФБ), в котором преодолены указанные выше недостатки.

Проблема заключается в том, что обработка больших объемов данных связана с существенными временными затратами, которые могут стать недопустимыми. Если это выявится на работающей системе, то устранить проблему будет не просто. Во-первых, задач обработки много. Во-вторых, если задачи сложные, то изменение настроек не помогает. В таком случае необходимо изменять алгоритмы обработки данных, т.е. переделывать ранее выполненную работу. А на это требуются дополнительные временные затраты и человеческие ресурсы. Следовательно, целесообразно оценивать время обработки данных на этапе проектирования, т.е. перед реализацией запросов с помощью метода КИФБ, что позволяет вовремя принять правильное решение. В этом случае адекватные математические модели являются незаменимым инструментом анализа. Данная модель также позволяет оценить эффективность разработанного метода КИФБ при изменении параметров системы в широком диапазоне.

В статье на примере запросов Q3 и Q17 из теста TPC-H [8] проанализированы процессы выполнения запросов в Spark. На основе этого анализа разработана стоимостная модель для оценки времени выполнения запросов к хранилищу данных, выполнена калибровка модели и получена оценка ее адекватности по результатам натурных экспериментов.

Анализ процессов выполнения запросов в SPARK

В табл. 1 приведены запрос Q3 из теста TPC-H (1-й столбец), его декомпозиция на подзапросы (поиск в одной таблице) и соединения результатов выполнения подзапросов (поиск в двух таблицах) (2-й столбец) [5]. После некоторых операторов select (C1, C1O1) строятся фильтры Блума [9,10], а в других

операторах (O1, L1) эти фильтры используются для фильтрации записей таблиц. На основе спецификаций из 2-го столбца была разработана программа на языке Scala [1].

Таблица 1

Запрос Q3	Подзапросы и соединения
select l_orderkey, sum(l_extendedprice*(1-l_discount)) as revenue, o_orderdate, o_shippriority **from** customer, orders, lineitem **where** c_mktsegment = '[SEGMENT]' and c_custkey = o_custkey and l_orderkey = o_orderkey and o_orderdate < date '[DATE]' and l_shipdate > date '[DATE]' **group by** l_orderkey, o_orderdate, o_shippriority **order by** revenue desc, o_orderdate;	*Подзапрос C1:* **select** c_custkey as kc1 **from** customer C_ **where** c_mktsegment = '[SEGMENT]'; Создание фильтра Блума b1 для атрибута C1.kc1; *Подзапрос O1:* **select** o_custkey as kc1, o_orderkey as ko1, o_orderdate as dt1, o_shippriority as sh1 **from** orders O_ **where** o_orderdate < date '[DATE]' **filter** o_custkey=b1; Соединение C1O1: **select** O1.ko1, O1.dt1, O1.sh1 **from** C1, O1 **where** C1.kc1=O1.kc1; Создание фильтра Блума b2 для атрибута O1.ko1; *Подзапрос L1:* **select** l_orderkey as ko1, l_discount as d1, l_extendedprice as e1 **from** lineitem L_ **where** l_shipdate > date '[DATE]' **filter** l_orderkey=b2; Соединение C1O1L1: **select** C1O1.ko1, C1O1.dt1, C1O1.sh1, L1.e1, L1.d1 **from** C1O1, L1 **where** C1O1.ko1=L1.ko1;

На рис. 1 представлен процесс обработки запроса Q3 в среде Spark, полученный после анализа результатов работы монитора [11].

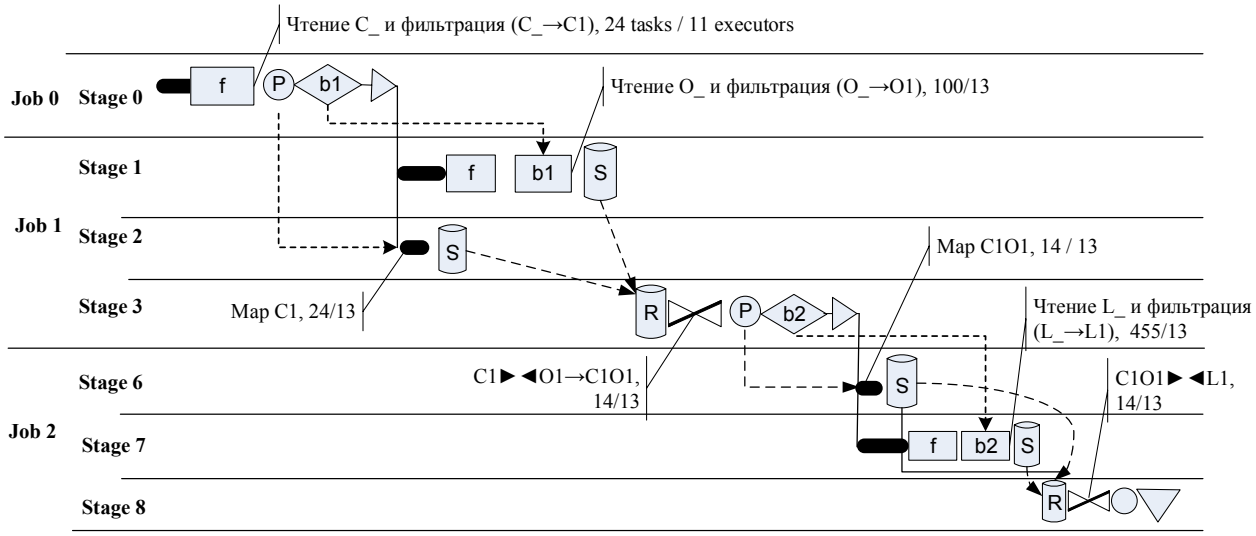

Рис. 1. Процесс выполнения запроса Q3 в среде Spark.

На рис. 2 приведены обозначения, принятые на рис. 1.

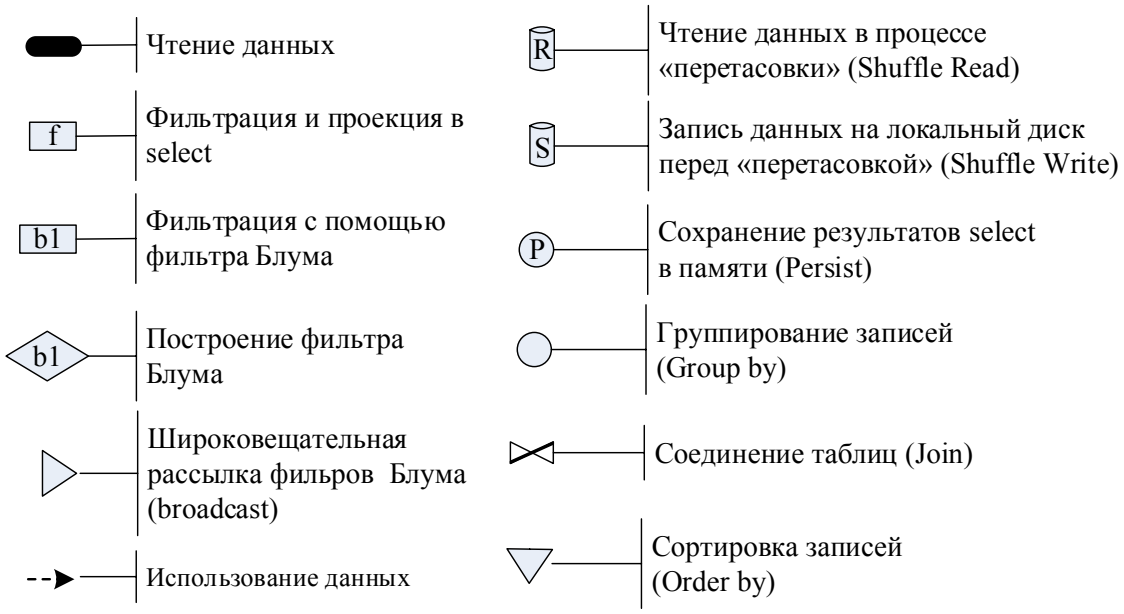

Рис. 2. Обозначения, принятые на рис. 1.

Из рис. 1 следует, что можно выделить два куста соединения таблиц. Первый куст связан со стадией Stage 3. Здесь выполняется соединение промежуточных таблиц (C1 и O1), полученных в результате обработки исходных таблиц C_ и O_ (стадии Stage 0 и Stage 1). Перед соединением таблицы C1 и O1 сериализуются, сохраняются на локальных дисках узлов (операции S на рис. 1), а затем выполняется их «перетасовка» [12] (shuffle – чтение с дисков и передача их по сети – операция R на рис. 1).

Второй куст связан со стадией Stage 8. На этой стадии выполняется соединение промежуточных таблиц (C1O1 и L1), полученных в результате соединения C1 и O1 и обработки исходной таблицы L_ (стадии Stage 3 и Stage 7). Здесь также выполняется «перетасовка» соединяемых таблиц.

В табл. 2 приведены запрос Q17 (с коррелированным подзапросом) из теста TPC-H (1-й столбец), его декомпозиция на подзапросы (поиск в одной таблице) и соединения результатов выполнения подзапросов (поиск в двух таблицах) (2-й столбец) [6].

На рис. 3 представлен процесс обработки запроса Q17 в среде Spark, полученный после анализа результатов работы монитора.

Здесь также можно выделить два куста соединения таблиц. Первый куст связан со стадией Stage 2. Выполняется соединение промежуточных таблиц (P1 и L1), полученных в результате обработки исходных таблиц P_ и L_ (стадии Stage 0 и Stage 2). Следует отметить, что Spark автоматически дублирует небольшую промежуточную таблицу P1 по всем узлам кластера (BroadcastExchange), хеширует ее в ОП узлов и выполняет соединение таблицы L1 с таблицей P1 «на лету» (BroadcastHashJoin) (Stage2).

Тем самым метод КИФБ позволяет сочетать применение фильтра Блума и дублирование небольших таблиц по узлам при выполнении одного запроса.

Таблица 2

Запрос Q17	Подзапросы и соединения
select sum(l_extendedprice)/7.0 as avg_yearly **from** lineitem, part **where** p_partkey = l_partkey and p_brand = '[BRAND]' and p_container = '[CONTAINER]' and l_quantity < (**select** 0.2 * avg(l_quantity) **from** lineitem **where** l_partkey = p_partkey);	*Подзапрос P1:* **select** p_partkey as pr1 **from** part P_ **where** p_brand = '[BRAND]' **and** p_container = '[CONTAINER]'; Создание фильтра Блума b1 для P1.pr1; *Подзапрос L1:* **select** l_partkey **as** pr1, l_extendedprice **as** e1, l_quantity **as** q1 **from** lineitem L_ **filter** L1.pr1=b1; *Соединение L2:* **select** L1.pr1,L1.e1,L1.q1 **from** L1, P1 **where** L1.pr1= P1.pr1; Создание фильтра Блума b2 для L2.pr1; *Подзапрос A1:* **Select** L2.pr1, 0.2*avg(L2.q1) **as** a1 **from** L2 **group by** L2.pr1 **filter** A1.pr1=b2 *Соединение A2:* **Select sum**(L2.e1)/7.0 **from** L2, A1 **where** L2. q1<A1.a1 **and** L2. pr1=A1.pr1

Второй куст связан со стадией Stage 6. На этой стадии выполняется соединение промежуточных таблиц (L2 и A1). Причем соответствующий подзапрос A1 реализуется в две стадии (Stage 3 и Stage 5), так как в нем присутствует группировка (group by).

В обоих кустах также выполняется «перетасовка» (shuffle) соединяемых таблиц.

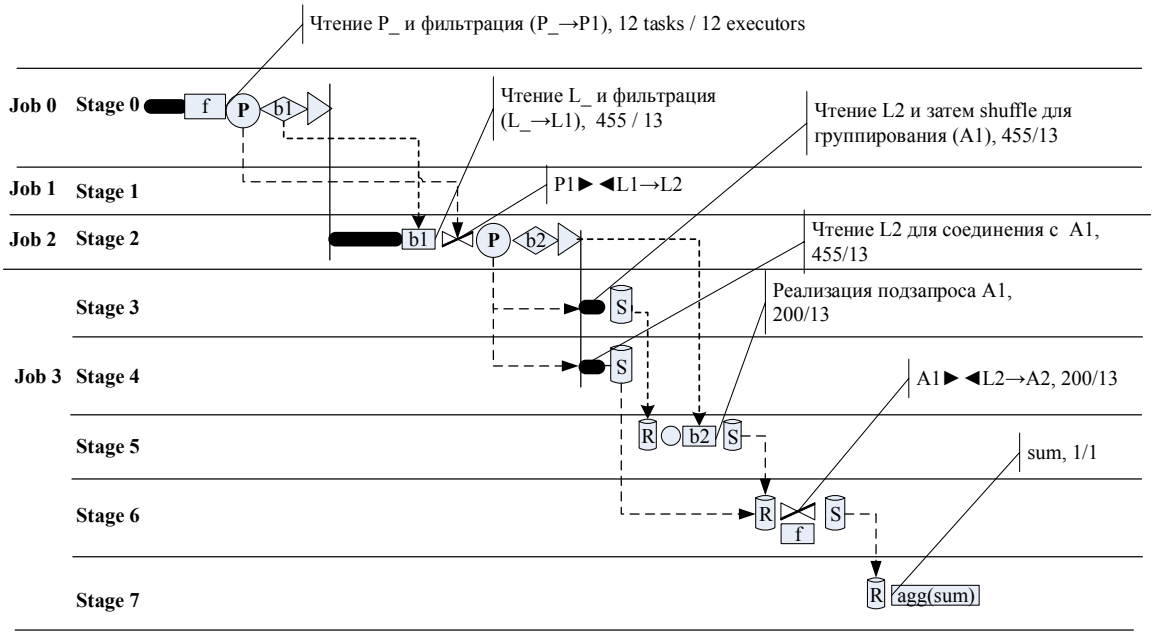

Рис. 3. Процесс выполнения запроса Q17 в среде Spark.

На основе анализа процессов можно выделить общие операции, которые используются в дальнейшем для разработки математической модели:

для куста – чтение таблиц куста, их соединение;

для таблицы куста – чтение, фильтрация, сохранение в памяти для дальнейшего использования (persists), построение или использование фильтра Блума, широковещательная рассылка фильтра Блума (после построения), соединение «на лету» (если предыдущая таблица небольшая), shuffle write (сохранение на локальных дисках для дальнейшего соединения);

для соединения – shuffle read (чтение с локальных дисков и «перетасовка» по сети), соединение (join), shuffle write (возможное сохранение на локальных дисках, если необходимо дальнейшее соединение с другими таблицами).

Разработка математической модели

На каждой стадии Spark порождает один или несколько параллельных процессов. Пример таких процессов приведен на рис. 4.

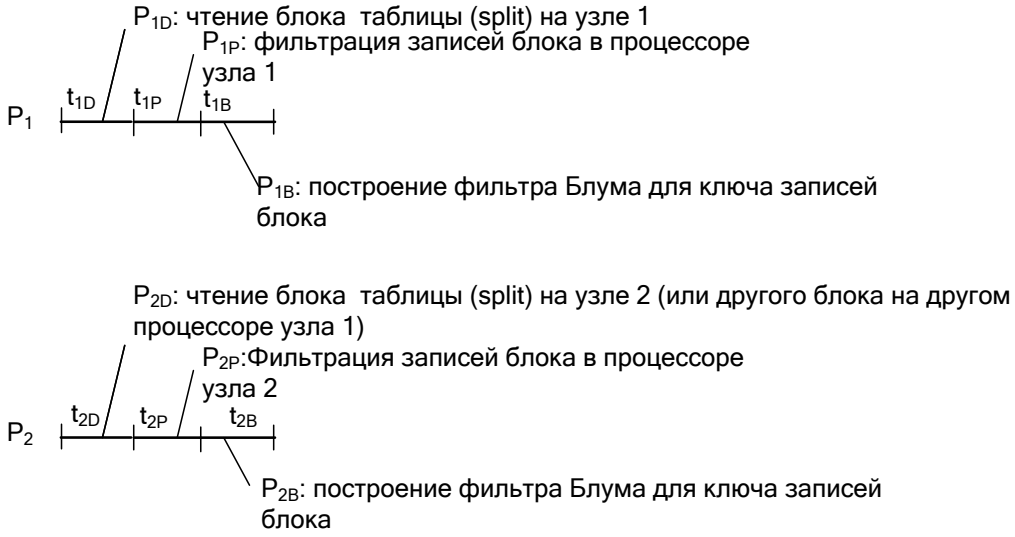

Рис. 4. Пример процессов, порождаемых Spark.

Здесь отрезками прямой обозначены интервалы определения процессов (начало и завершение). Над отрезками указаны продолжительности временных интервалов (t_{ix}) – время потребления ресурса. Мы будем рассматривать средние значения. Процесс может порождать другой процесс (через промежуток времени t_{ix}). Например, для узла 1 цепочка порождаемых процессов выглядит так: $P_{1D} \rightarrow P_{1P} \rightarrow P_{1B}$. Как видно из рисунка, пара одноименных процессов, выполняемых на разных узлах (или процессорах того же узла), образует группу *параллельных подобных процессов* (ППП). Здесь можно выделить три группы: (P_{1D}, P_{2D}), (P_{1P}, P_{2P}), (P_{1B}, P_{2B}). Понятие подобных процессов, в другом контексте, используется в имитационном моделировании [13].

Группу ППП будем также обозначать отрезком прямой (рис. 5), который соответствует интервалу определения группы.

Родительские ППП порождают дочерние параллельные подобные процессы. Например, дочерние ППП P_P порождены родительскими ППП P_D и т.д. Объединение групп ППП будем называть *связанными параллельными процессами* (СПП). Например, совокупность ППП P_D, P_P, P_B образует СПП с идентификатором P (рис. 5).

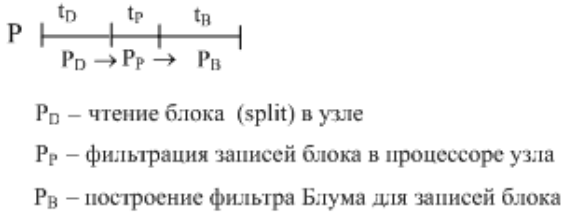

Рис. 5. Обозначение параллельных подобных процессов.

Для удобства СПП будем обозначать одним отрезком прямой, который соответствует *интервалу СПП*. Продолжительность этого интервала равна времени между началом и окончанием активности всех параллельных процессов, входящих в объединение. Это время будем называть временем выполнения связанных параллельных процессов. Каждый интервал СПП будем снабжать идентификатором (например, P на рис. 5). Часто экземпляр СПП соответствует задаче, выполняемой в слоте Исполнителя (executor).

По результатам анализа процессов выполнения запросов в Spark была разработана обобщенная математическая модель. Она относится к классу стоимостных моделей [14]. На рис. 6 приведено описание процессов выполнения подзапросов и соединений, соответствующих одному кусту исходного запроса. Каждый отрезок прямой на рис. 6 соответствует СПП.

Ниже перечислены связанные параллельные процессы (СПП), представленные на рис. 6.

1. СПП R_i. Чтение и обработка таблиц измерений, построение фильтров Блума ко ключу (bloomfilter(k_i)).

2. СПП A. Сборка фильтров Блума в программе Драйвера (collect), объединение по ИЛИ, их широковещательная рассылка по узлам (broadcast).

3. СПП R_F. Чтение и обработка таблицы фактов, фильтрация записей с помощью фильтров Блума (fk_i – внешний ключ таблицы фактов).

Иногда перед применением фильтров Блума выполняется операция группирования (group by) для таблицы фактов (см. табл. 2, подзапрос A1).

Далее пункты 4, 5 выполняются, если L>0 (L – число небольших таблиц).

4. СПП B. Широковещательная рассылка отфильтрованных таблиц измерений, размер которых не превышает некоторого граничного значения V_M.

5. СПП C. Соединение таблицы фактов с таблицами измерений 1÷L в ОП (HashJoin).

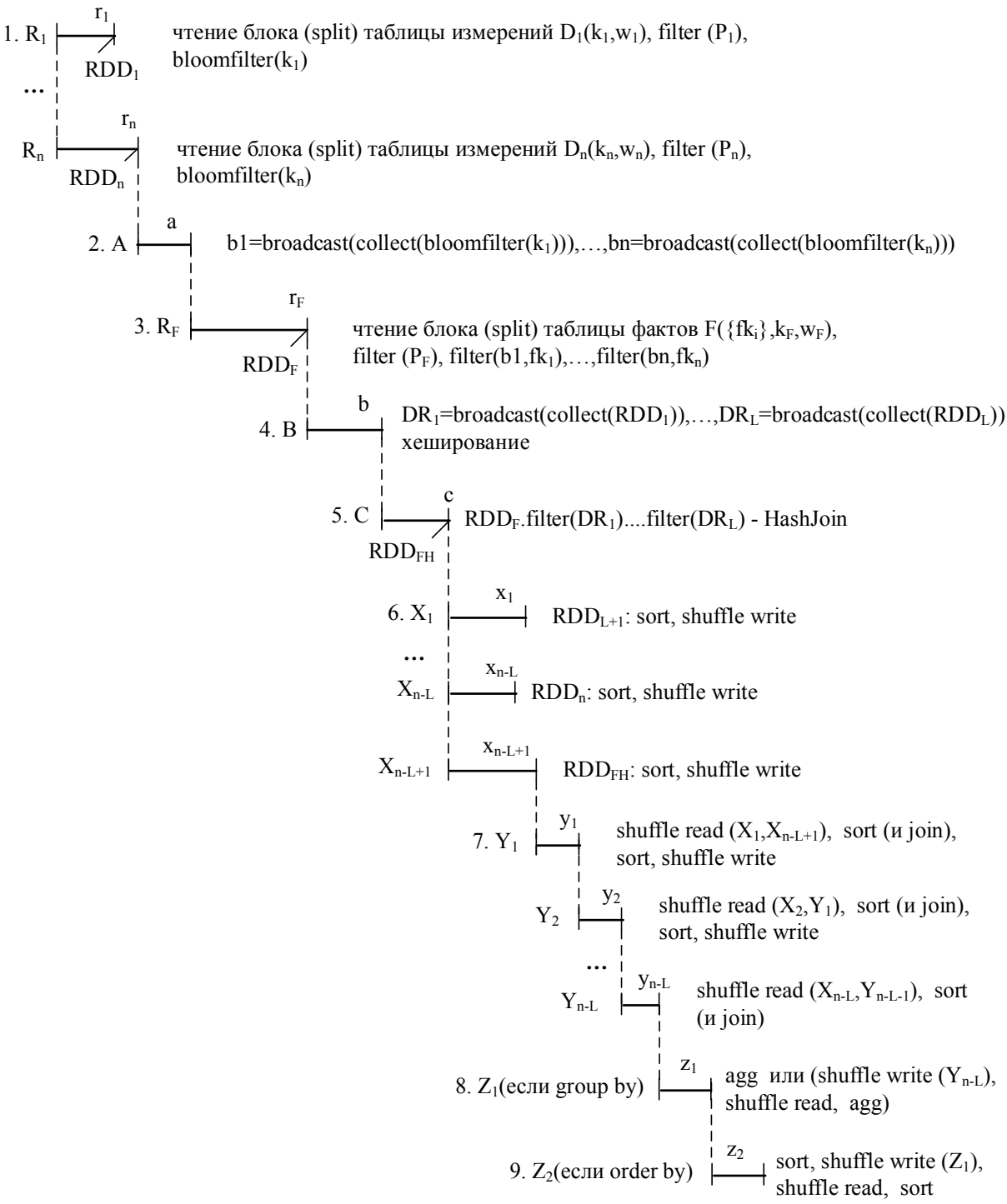

Рис. 6. Описание процессов реализации подзапросов и соединений, соответствующих кусту исходного запроса, для метода КИФБ.

Далее пункты 6, 7 выполняются, если L < n.

6. СПП X_i (i=1...n-L+1). Сортировка на стороне map каждого раздела таблицы измерений (RDD_{L+1}...RDD_n) или таблицы фактов (RDD_{FH}) по ключу соединения, сохранение отсортированных разделов в локальной файловой системе (Shuffle Write).

7. СПП Y_i (i=1...n-L). Попарное соединение таблицы фактов и таблиц измерений.

Далее пункты 8, 9 выполняются, если в исходном запросе указаны конструкции group by (Z_1) и order by (Z_2).

8. СПП Z_1. Группирование в конце выполнения запроса.

Пункт 9 выполняется, если указана конструкция order by.

9. СПП Z_2. Сортировка в конце выполнения запроса.

Таблица, получаемая в результате выполнения подзапросов и соединений куста исходного запроса, может служить измерением в следующем кусте.

В табл. 3 приведено соответствие стадий 1-го и 2-го кустов соединения таблиц в запросе Q3 (рис. 1) связанным параллельным процессам модели (рис. 6).

Таблица 3

Стадии	Связанные параллельные процессы модели для запроса Q3
stage0	СПП R_1 – чтение C_ из HDFS и фильтрация (C_→C1), построение фильтра Блума b1
	СПП A – сборка и рассылка фильтра Блума b1
stage1	СПП R_F – чтение O_ из HDFS, фильтрация, фильтрация с помощью фильтра Блума b1 (O_→O1)
	СПП X_2 – сортировка на стороне map и shuffle write O1
stage2	СПП X_1 – сортировка на стороне map и shuffle write C1
stage3	СПП Y_1 – shuffle read, соединение C1 и O1 (C1►◄ O1→C1O1)
	СПП R_1 – **начало 2-го куста**, C1O1 находится в ОП, построение фильтра Блума b2
	СПП A – сборка и рассылка фильтра Блума b2
stage4,5	пропускаются
stage6	СПП X_1 – сортировка на стороне map и shuffle write (C1O1)
stage7	СПП R_F – чтение L_ из HDFS, фильтрация, фильтрация с помощью фильтра Блума b2 (L_→L1)
	СПП X_2 – сортировка на стороне map и shuffle write L1
stage8	СПП Y_1 – shuffle read, соединение C1O1►◄L1
	СПП Z_1 – group by
	СПП Z_2 – order by

В табл. 4 приведено соответствие стадий 1-го и 2-го кустов соединения таблиц в запросе Q17 (рис. 3) связанным параллельным процессам модели (рис. 6)

Таблица 4

Стадии	Связанные параллельные процессы модели для запроса Q17
stage0	СПП R_1 – чтение P_ из HDFS и фильтрация (P_→P1), построение фильтра Блума b1
	СПП A – сборка и рассылка фильтра Блума b1
stage1	пропускается
stage2	СПП R_F – чтение L_ из HDFS, фильтрация, фильтрация с помощью фильтра Блума b1 (L_→L1)
	СПП B,C – сборка и широковещательная рассылка P1, соединение P1►◄ L1→L2 (HashJoin)

Продолжение табл. 4

	СПП R_1 – **начало 2-го куста**, L2 в ОП (таблица измерения), построение фильтра Блума b2
	СПП A – сборка и рассылка фильтра Блума b2
stage3	СПП R_F (с группированием) – L2 в ОП (таблица фактов), shuffle write L2
stage5	shuffle read L2 (таблица фактов), группирование, фильтрация с помощью фильтра b2 (A1)
	СПП X_2 – сортировка на стороне map и shuffle write A1
stage4	СПП X_1 – сортировка на стороне map и shuffle write (L2 – таблица измерений)
stage6	СПП Y_1 – shuffle read, соединение (A1►◄L2→A2), shuffle write
stage7	shuffle read, агрегирование (sum)

Ограниченный объем статьи не позволяет привести все формулы для расчета интервалов 1 – 9 СПП. В качестве примера приведем формулы для расчета интервала «СПП R_i». В формулах (1) – (7) использованы следующие обозначения: N – число узлов (рабочих) в кластере; NC – общее число ядер CPU в кластере (число слотов Исполнителей, число физических ядер); V_S – размер блока split (байт); V_{Di}, Q_{Di} – объем (сжатый) и число записей i-й таблицы измерений (i = 1,…, n); P_i – вероятность, что запись удовлетворяет условию поиска по i-й таблице измерений; μ_{RH} – интенсивность чтения данных из файловой системы HDFS (байт/с); τ_d – процессорное время десериализации (развертывания) задачи в слоте Исполнителя; τ_f – процессорное время фильтрации на запись; τ_b – процессорное время записи/чтения записи из фильтра Блума.

Число слотов (задач, tasks), необходимых для обработки i-й таблицы измерений, равно

$$N_{Si} = \lceil V_{Di} / V_S \rceil . \quad (1)$$

Число записей на слот можно рассчитать по формуле

$$Q_{Si} = Q_{Di} / N_{Si} . \quad (2)$$

Объем split для i-й таблицы измерений равен

$$V_{Si} = \begin{cases} V_S, \textit{ если } N_{Si} \geq 2, \\ V_{Di}, \textit{ если } N_{Si} = 1 \end{cases} \quad (3)$$

Рассчитаем время выполнения одной задачи, связанной с обработкой записей i-й таблицы измерений:

$$r_i = \tau_d + V_{Si} / \mu_{RH} + \tau_f Q_{Si} + \tau_b P_i Q_{Si} . \quad (4)$$

Но в слоте Исполнителя могут последовательно выполняться несколько задач. Пусть в слот спланирована задача, связанная с m-й таблицей измерений:

$$m : r_m = \max \{r_i\} . \quad (5)$$

Блоки (splits) таблиц хранилища данных распределены по узлам кластера равномерно (равновероятно).

Тогда в тот же слот будут спланированы остальные задачи таблиц измерений, каждая из которых обрабатывает один split таблицы, с вероятностью

$$\frac{1}{N} \cdot \frac{1}{NC / N} = \frac{1}{NC} \ . \qquad (6)$$

Получим время обработки таблиц измерений:

$$r = r_m + \frac{N_{Sm} - 1}{NC} r_m + \sum_{\substack{i=1 \\ i \neq m}}^{n} \frac{N_{Si}}{NC} r_i = (1 - \frac{1}{NC}) r_m + \frac{1}{NC} \sum_{i=1}^{n} N_{Si} r_i \ . \qquad (7)$$

Калибровка параметров модели и анализ ее адекватности

Математическая модель, реализующая процессы, представленные на рис. 6, была разработана на языке Python. Для калибровки модели был использован стенд, включавший виртуальный кластер. Кластер состоял из 8 узлов, на одном из которых были инсталлированы управляющие службы HDFS, Hive, Spark, Yarn [15]. Основные характеристики каждого виртуального узла следующие: один двуядерный процессор, 8 GB оперативной памяти, 200 GB SSD диск, ОС Ubuntu 14.16. Результаты натурных экспериментов разбиты на два подмножества.

В первое подмножество входили оценки времени выполнения десяти запросов.

Пять запросов Q3 (табл. 1) со следующими параметрами наполнения базы данных:

SF = 500 (NBF = 40 млн.);

SF = 250 (NBF = 50 млн.);

SF = 100, SF = 50,

SF = 10 (NBF = 15 млн.),

где NBF – предполагаемое число элементов в фильтрах Блума b1 и b2 [9,10].

Пять запросов Q17 (табл. 2) со следующими параметрами наполнения базы данных:

SF = 500, SF = 250,

SF = 100, SF = 50,

SF = 10 (NBF = 15 млн.).

Соответствующие оценки времени выполнения запросов использовались для калибровки параметров модели методом наименьших квадратов (МНК) с использованием градиентного спуска. Калибруемые параметры модели и диапазоны их изменения приведены в табл. 5.

Таблица 5

Калибруемый параметр модели	Нижняя граница	Верхняя граница	Оптимальное значение
τ_f – процессорное время фильтрации на запись	1.0E-06 с	1.0E-05 с	1.14E-06 с
τ_b – процессорное время записи/чтения записи из фильтра Блума	1.0E-08 с	1.0E-07 с	2.07E-08 с
τ_s – время сортировки на запись	1.0E-08 с	1.0E-07 с	2.11E-08 с
τ_d – процессорное время десериализации (развертывания) задачи в слоте Исполнителя	1.0E-06 с	1.0E-04 с	7.59E-05 с
τ_h – время хеширования (для дальнейшего сравнения или агрегации) на запись	1.0E-08 с	1.0E-06 с	4.27E-07 с
KS – коэффициент, учитывающий влияние сериализации на объем передаваемых данных при shuffle	0.5	1.5	0.81
μ_{RH} – интенсивность чтения данных из файловой системы HDFS	20.0 МБ/с	50.0 МБ/с	44.14 МБ/с
μ_{WL} – интенсивность записи данных в локальную файловую систему LFS	50.0 МБ/с	200 МБ/с	61.36 МБ/с
μ_{N1} – интенсивность передачи данных через коммутатор сети	100 МБ/с	500 МБ/с	185.7 МБ/с

Поиск оптимальных значений калибруемых значений модели выполнялся следующим образом [16]: во внешнем цикле случайным образом выбиралась точка в 9-мерном параллелепипеде (9 – число калибруемых параметров), во внутреннем цикле численным методом осуществлялась минимизация функции ошибки моделирования с использованием градиентного спуска. Функция ошибки – это сумма квадратов разности экспериментальных и модельных значений времени выполнения 10 запросов. Так как минимумов функции ошибки может быть много и возможны выходы за пределы диапазонов, внешний цикл повторялся 100 раз. Полученные оптимальные значения калибруемых параметров модели приведены в последнем столбце табл. 5. Сумма квадратов отклонений значений времени, полученных с использованием модели, от экспериментальных значений составила 27079 для 10 запросов. Во второе подмножество экспериментальных результатов (40 точек) входили оценки времени выполнения стадий запросов – стадии 0, 1, 2, 3, 6, 7, 8 запроса Q3 (табл. 3) со следующими параметрами наполнения базы данных: SF = 500 (NBF = 40 млн.), SF = 250 (NBF = 50 млн.), SF = 100, SF = 50, SF = 10 (NBF = 15 млн.); и стадии 0, 2, (3 + 5), 4, (6 + 7) запроса Q17 (табл. 4) с параметром наполнения базы данных SF = 500 (NBF = 15 млн.). Модельные значения времени выполнения стадий оценивались для полученных оптимальных значений откалиброванных параметров (см. табл. 5).

Все модельные и экспериментальные значения времени выполнения запросов и стадий из двух подмножеств представлены в виде точек «модель – эксперимент» на рис. 7 (50 точек).

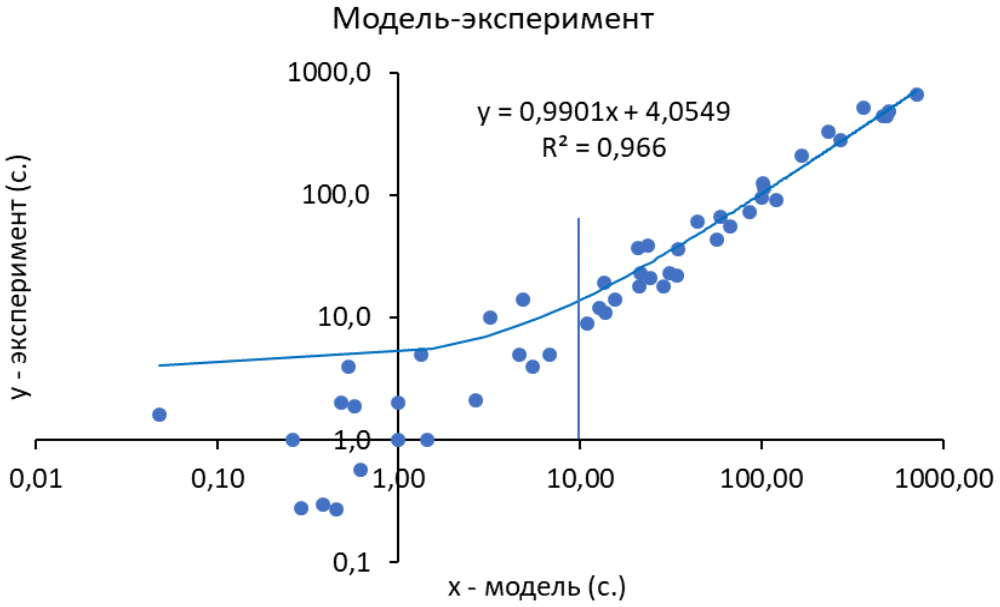

Рис. 7. Точечная диаграмма «модельное значение времени выполнения запросов и стадий – экспериментальное значение времени».

По осям x и y выбран логарифмический масштаб: $x1 = \log_{10}x$, $y1 = \log_{10}y$. В общем случае экспериментальные значения y – это случайные величины. На рис. 7 приведена регрессионная зависимость $y = 0.99x + 4.0 \approx x + 4$. В координатах x1 и y1 эту зависимость можно представить в следующем виде: $y1 = \log_{10}(10^{x1} + 4)$. При достаточно больших x1 имеем y1 = x1 и, следовательно, y = x. При $x1 \to -\infty$, $y1 \to \log_{10}4$ (горизонтальная асимптота). Коэффициент достоверности аппроксимации экспериментальных данных регрессионной зависимостью близок к 1 ($R^2 = 0.966$), что свидетельствует о достоверности моделирования при достаточно больших модельных значениях времени (в этом случае y = x). Из графика на рис. 7 видно, что при модельном значении больше 10 с точность моделирования хорошая (точки концентрируются относительно прямой y = x). Относительная погрешность моделирования ($\Delta = 100 \cdot |T_{экс} - T_{мод}|/T_{экс}$) для точек, расположенных правее прямой x = 10 (31 точка), имеет следующее распределение: процент ошибки $\Delta \le 10\%$ имеют 35% точек; $10\% < \Delta \le 20\%$ – 19% точек; $20\% < \Delta \le 30\%$ – 23% точек; $30\% < \Delta \le 40\%$ – 13% точек; $\Delta > 40\%$ – 10% точек.

Заключение

На примере запросов Q3 и Q17 из теста TPC-H проанализированы процессы выполнения запросов в распределенной параллельной системе обработки данных Spark (рис. 1 и 3). Разработана математическая модель этих процессов, основу которой составляют подмодели связанных параллельных процессов (рис. 6). Разработанная модель не ограничивается запросами Q3 и

Q17 и может быть использована для прогнозирования времени выполнения других запросов к хранилищу данных.

По результатам натурных экспериментов (всего 50 точек) выполнена калибровка параметров модели и получена оценка ее адекватности. Коэффициент достоверности аппроксимации регрессионной зависимостью составил $R^2 = 0.966$, что свидетельствует о хорошей точности модели при достаточно больших значениях модельного времени. Показано, что точки со значением модельного времени больше 10 с концентрируются относительно прямой $y = x$ (рис. 7). При этом 77% этих точек имеют относительную погрешность моделирования $\Delta < 30\%$. Этого достаточно для использования модели на этапе проектирования распределенной параллельной системы, где выполняется сравнение и выбор варианта.

С увеличением объема обрабатываемых данных (SF = 10, ..., 500) точность моделирования растет (рис. 7). Это согласуется с результатами, полученными в [14].

ЛИТЕРАТУРА

1. *Григорьев Ю.А., Пролетарская В.А., Ермаков Е.Ю.* Экспериментальная проверка эффективности метода доступа к хранилищу данных на платформе SPARK с каскадным использованием фильтра Блума // Информатика и системы управления. – 2017. – № 3. – С. 3-16.
2. *Karau Holden et al.* Learning spark: lightning-fast big data analysis // O'Reilly Media, Inc., 2015.
3. *Karau Holden and Rachel Warren*. High Performance Spark: Best Practices for Scaling and Optimizing Apache Spark // O'Reilly Media, Inc., 2017.
4. *Jaqueline Joice Brito, Thiago Mosqueiro, Ricardo Rodrigues Ciferri, Cristina Dutra de Aguiar Ciferri*. Faster cloud Star Joins with reduced disk spill and network communication. // ICCS 2016. The International Conference on Computational Science. – 2016. –Vol. 80. – P. 74-85.
5. *Григорьев Ю.А., Пролетарская В.А., Ермаков Е.Ю.* Метод доступа к хранилищу данных по технологии SPARK с каскадным использованием фильтра Блума // Информатика и системы управления. – 2017. – № 1. – С. 3-14.
6. *Пролетарская В.А.* Каскадное использование фильтра Блума при реализации SQL-запроса с коррелированным подзапросом на платформе параллельной обработки данных SPARK // Сборник научных трудов XII Международной научно-практической конференции «Конвергентные когнитивно-информационные технологии. Современные информационные технологии и ИТ-образование». – 2017. – С. 259-264.
7. *Grigoriev Yu.A., Proletarskaya V.A., Ermakov E.Yu., Ermakov O.Yu.* Efficiency Analysis of the access method with the cascading Bloom filter to the data warehouse on the parallel computing platform // Journal of Physics: Conference Series. – IOP Publishing, 2017. – Vol. 913, № 1. – P. 012011.
8. TPC-H. http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.2.pdf .
9. *Bloom B.H.* Space/time trade-offs in hash coding with allowable errors //Communications of

the ACM. – 1970. – Vol. 13. – №. 7. – P. 422-426.
10. *Tarkoma S., Rothenberg C.E., Lagerspetz E.* Theory and practice of bloom filters for distributed systems. IEEE Communications Surveys and Tutorials. – 2012. – 14(1). – P. 131-155.
11. *Andrew Or.* Understanding your Apache Spark Application Through Visualization. https://databricks.com/blog/2015/06/22/understanding-your-spark-application-through-visualization.html .
12. *Григорьев Ю.А., Пролетарская В.А.* Метод ранней материализации доступа к хранилищу данных по технологии MapReduce // Информатика и системы управления. – 2015. – № 3. – С. 3-14.
13. *Черненький В.М.* Теоретические основы построение имитационного процесса. Учебное пособие по дисциплине «Описание параллельных процессов». – М., 2015. https://elibrary.ru/download/elibrary_32161665_40023934.pdf.
14. *Григорьев Ю.А*. Об использовании стоимостной модели для прогнозирования времени выполнения запросов к базе данных // Информатика и системы управления. – 2018. – № 1. – С. 3-15.
15. *Vavilapalli Vinod Kumar et al.* Apache hadoop yarn: Yet another resource negotiator // Proceedings of the 4th annual Symposium on Cloud Computing. ACM. – 2013 – P. 5.
16. *Григорьев Ю.А., Бурдаков А.В., Пролетарская В.А., Устимов А.И.* Анализ процесса выполнения запросов к хранилищу данных по методу соединения реплик с несколькими фрагментами в MapReduce // Динамика сложных систем — XXI век. – 2018. – №1. – С. 62-76.

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