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)

# АНАЛИЗ ВРЕМЕНИ СОЕДИНЕНИЯ ТАБЛИЦ В СТРОЧНОЙ ПАРАЛЛЕЛЬНОЙ СИСТЕМЕ БАЗ ДАННЫХ И ПО ТЕХНОЛОГИИ MAPREDUCE

Анализ публикаций показал, что для реализации запросов к структурированным данным преимущество отдается параллельным СУБД. Технология MapReduce (MR) рассматривается как дополнение к технологии СУБД. В статье на примере задачи Join мы попытались выяснить, как поведут себя строчная параллельная СУБД и MR-система Hadoop, если варьировать параметры, которые в проведенных другими авторами экспериментах оставались постоянными или были другими. Ранее были разработаны детальные модели процессов соединения таблиц в строчной параллельной СУБД и MR-системе. В статье приведены результаты вычислительных экспериментов на этих моделях. Модели были настроены на разные схемы масштабирования для MR (число узлов), СУБД (объем данных в узле) и фрагментацию соединяемых таблиц по первичному ключу. Варьировались следующие параметры: селективность выбираемых данных, число сортируемых результирующих записей, мощность атрибута группирования. Результаты моделирования показали, что при увеличении объема хранимых данных параллельная СУБД проигрывает MR-системе по производительности, начиная с некоторых пороговых значений.

***Ключевые слова:*** СУБД**,** SQL, технология MapReduce, запрос на соединение таблиц, оценка времени выполнения запроса, сравнение времени.

## Введение

Известно [11], что изначально технология MapReduce (MR) ориентировалась на обработку огромных объемов неструктурированных текстовых данных на компьютерных кластерах с количеством узлов более 1000.

В статье [2] на конкретных тестовых примерах показано, что при обработке структурированных данных система MR проигрывает по производительности параллельным СУБД. Как отмечается в [3], различия в производительности между MR и СУБД объясняются различными факторами. При использовании СУБД не требуется выполнять разбор записей в прикладной программе. Здесь применяются эффективные алгоритмы сжатия данных, конвейеризация, планирование, поколоночное хранение данных (в колоночных СУБД). Эксперименты проводились на

100 узлах, но авторы статьи 12] полагают, что те же относительные показатели производительности сохранились бы и на кластере с 1000 узлов (хотя оговариваются, что не могут проверить это утверждение). В то же время MR выигрывает по отказоустойчивости у параллельных СУБД [2, 4]. Авторы статьи [2] в другой своей публикации [3] подтверждают предыдущие результаты и приходят к выводу, что системы MR больше похожи на системы извлечения-преобразования-загрузки (extract-transform-load, ETL). Они быстро загружают и обрабатывают в заранее непредвиденном режиме данные большого объема. В этом качестве технология MR дополняет технологию СУБД, а не конкурирует с ней.

В [3] также отмечается, что многие СУБД поддерживают функции, определяемые пользователями (user-defined functions, UDF). Они могут выступать в качестве аналогов функций map и reduce, используемых в системах MR. Эти функции можно применять для реализации операций, которые не так легко выражаются средствами SQL. В [5] предлагается новый подход к реализации UDF, называемый SQL/MapReduce (SQL/MR) и позволяющий преодолеть многие из ограничений традиционного механизма UDF.

В проекте HadoopDB [4] (гибрид MR и СУБД) основная идея состоит в использовании MR в качестве координатора и коммуникационного слоя над несколькими узлами, в которых выполняются экземпляры одноузловой СУБД PostgreSQL. Запросы представляются на языке SQL, транслируются в MR расширенными существующими средствами (Hadoop Hive), и как можно б*о*льшая часть работы передается в высокопроизводительные одноузловые СУБД. На тех же тестовых запросах ([2]) демонстрируется, что HadoopDB превосходит по производительности Hadoop, но уступает параллельным СУБД.

Как следует из рассмотренных публикаций, для реализации запросов к структурированным данным преимущество отдается параллельным СУБД. Технология MR рассматривается как дополнение к технологии СУБД. В основном этот вывод делается по результатам натурных экспериментов, опубликованным в работе [2]. Мы не ставим под сомнение сами результаты тестирования. Но анализ архитектуры стенда и тестируемых запросов вызывает ряд вопросов: 1) эксперименты проводились с числом узлов от 1 до 100. Авторы [2] утверждают, что на 100 узлах две параллельные СУБД справляются с разными аналитическими задачами быстрее, чем MR. Приведены примеры параллельных систем баз данных с числом узлов меньше 100 и объемом данных 1-2 петабайтов. Но следует отметить, что общий объем данных мало влияет на время выполнения запроса к базе данных. Важна селективность данных, связанная с запросом. А она во многих тестируемых примерах была довольно низкой (задача Selection, задача Join и др.); 2) почти во всех тестируемых запросах SELECT не выполнялась такая важная операция как упорядочивание результирующих записей. Только в задаче Join [2, п. 4.3.4] использовалась конструкция "ORDER BY" и то с одной записью в результирующей таблице (LIMIT 1); 3) в задаче Join таблица UserVisits фрагментировалась не по первичному ключу, а по атрибуту соединения destURL. Это дает дополнительное преимущество параллельным СУБД, так как позволяет избежать межмашинного обмена при выполнении запроса на соединение таблиц (не вы-

полняется операция SHUFFLE). Но такая фрагментация влияет на время выполнения других запросов. Например, при поиске по другим атрибутам время выполнения запроса может увеличиться, так как возможен межмашинный обмен.

В данной статье на примере задачи Join из набора тестов [2] мы попытались выяснить, как поведут себя строчная параллельная СУБД и MR-система Hadoop, если варьировать параметры, которые в экспериментах [2] оставались постоянными или были другими. Это отличные схемы масштабирования для MR (число узлов) и СУБД (объем данных в узле), селективность поисковых атрибутов, сортировка всех результирующих данных, увеличение мощности атрибута группирования вместе с ростом объема хранимых данных, фрагментация таблиц по первичному ключу.

Выбор СУБД был связан с тем, что многие коммерческие параллельные СУБД – строчные (Teradata, DB2 и др.). А наиболее популярной MR-системой является Hadoop – проект с открытыми исходными текстами, выполняемый Yahoo! и Apache Software Foundation [3]. Выбор задачи Join 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). Конечно, реализовать натурный эксперимент с 1000 и более узлами и большим объемом данных крайне затруднительно – существенно возрастает стоимость стенда, а также увеличивается время натурного моделирования и обработки результатов.

В работах [7, 8] дан подход к оценке времени соединения таблиц в параллельных СУБД с использованием методов Nested Loop Join и Hash Join на основе преобразования Лапласа-Стилтьеса (ПЛС). Но предлагаемые в этих статьях модели не учитывают ряд важных параметров и некоторые особенности обработки запросов в СУБД. В работах [9, 10] разработаны более детальные модели процессов соединения таблиц в строчной параллельной СУБД и MR-системе. С целью обеспечить адекватность моделей в аналитические выражения был введен параметр времени выполнения Короткой Логической Операции Алгоритма (КЛОА), позволивший выполнить калибровку моделей по результатам натурных экспериментов, описанных в [2]. В данной работе приведены результаты вычислительных экспериментов на этих моделях и проанализированы зависимости среднего времени выполнения задачи Join для параллельной СУБД и MR-системы.

## Сравнение среднего времени соединения таблиц в параллельной СУБД и в базе NoSQL по технологии MapReduce

Для калибровки моделей [9, 10] были использованы результаты натурных экспериментов, приведенные для задачи Join в [2, п. 4.3.4]. Эти результаты соответствуют выполнению следующего запроса:

```
SELECT sourceIP, AVG(pageRank) as avgPageRank, SUM(adRevenue) as totalRevenue
FROM Rankings AS R, UserVisits AS UV
WHERE R.pageURL = UV.destURL AND
UV.visitDate BETWEEN Date('2000-01-15') AND Date('2000-01-22')
GROUP BY UV.sourceIP
ORDER BY totalRevenue DESC LIMIT 1;
```

Этот запрос соответствует рассмотренному в предыдущих работах предложению SELECT [9, 10, см. (2), (1)]: R2-Rankings AS R – ранг страницы; R1-UserVisits AS UV – обращение пользователя к странице; R1.A1X-sourceIP – IP-адрес пользователя; f1(атр1)-AVG(pageRank) – средний ранг страницы; f2(атр2)-SUM(adRevenue) – суммарная выручка; q = 2; R1.A11-UV.destURL – страница, к которой обратился пользователь; R2.A21-R.pageURL – адрес страницы, условие по R1-UV.visitDate BETWEEN Date('2000-01-15') AND Date('2000-01-22'), условие по R2 – нет, Y=1 (LIMIT).

В табл. 1 приведены некоторые данные, полученные на основании анализа характеристик устройств и результатов измерений, приведенных в [2, п. 4.3.4] для задачи Join. Они были использованы для калибровки моделей. В табл. 2 приведены исходные данные, использованные в процессе моделирования.

*Таблица 1*

| Параметр | Значение | Примечание |
|---|---|---|
| **Технические средства** | | |
| | | Узел: один процессор Intel Core 2 Duo; 2,40 Ггц; 4 Гбайтов основной памяти; два 250-Гбайтных диска SATA-I; ОС Red Hat Enterprise Linux 5 (версия ядра 2.6.18). |
| | 50 | Сеть: максимальное число портов коммутатора Cisco Catalyst 3750E-48TD. |
| | 9 | Максимальное число коммутаторов в кольце. |
| **Параллельная строчная СУБД** | | |
| $T_{DBMS}$ | 29 с | Среднее время выполнения запроса при n≥10 (практически не зависит от числа узлов в кластере). |
| **MapReduce (Hadoop)** | | |
| $T_{F1} = T_{M1}+T_{R1}$ | 1435 с | Среднее время выполнения фазы 1 [2] (практически не зависит от числа узлов в кластере). |
| $T_{M11}$ | 600 с | Фаза 1 (map): чтение с диска таблиц UserVisits и Rankings. |
| $T_{M12}$ | 300 с | Фаза 1 (map): разбиение, разбор и десериализация различных атрибутов. |
| $T_{F2} = T_{M2}+T_{R2}$ | 24.3 с | Среднее время выполнения фазы 2 [2] (практически не зависит от числа узлов в кластере). |
| $T_{F3} = T_{M3}+T_{R3}$ | 12.7 с | Среднее время выполнения фазы 3 [2] (практически не зависит от числа узлов в кластере). |

*Таблица 2*

| Параметр | Значение | Примечание |
|---|---|---|
| **Параметры базы данных и запроса** [2, п. 4.3] | | |
| $V_1$ | 20 Гбайт | Объем таблицы UserVisits (R1) на узел. |
| $Q_1$ | $155 \cdot 10^6$ | Число записей в таблице UserVisits (R1) на узел. |
| $V_2$ | 1 Гбайт | Объем таблицы Rankings (R2) на узел. |
| $Q_2$ | $18 \cdot 10^6$ | Число записей в таблице Rankings (R2) на узел. |
| $p_1$ | $134000/Q1 = 8.6 \cdot 10^{-4}$ | Доля записей таблицы UserVisits, удовлетворяющих условию поиска, используется кластеризованный индекс на столбце UserVisits.visitDate. |
| $p_2$ | 1 | Так как в запросе нет условия поиска по таблице Rankings. |
| IA1X | 2,5·106 | Мощность атрибута sourceIP (R1.A1X ) [2, п. 4.3.3]. |
| lA | 24 байт | Длина записи (sourceIP, AVG(pageRank), SUM(adRevenue)), т.е. (R1A1X, agr1, agr2). |

*Продолжение табл. 2*

| | | |
|---|---|---|
| **Параллельная строчная СУБД** | | |
| $V_H$, $V_S$ | 512 Мбайт | Размер области хеширования в ОП, размер области памяти, отведенной под сортировку. Эти величины принимаются равными размеру буферного пула строчной СУБД. |
| $\mu_{DR1}$ | 20 Гбайт / 390 с = = 51 Мбайт/с | Пропускная способность файловой системы СУБД на чтение с учетом сжатия данных исходных таблиц [2, п. 4.3.3, рис. 8, Nodes 10, 25, 50, 100]. |
| $\mu_{DR1}$ | 51/2 = 25.5 Мбайт/с | Пропускная способность файловой системы СУБД на чтение без сжатия промежуточных таблиц [2, п. 4.1.1, СУБД-X – время увеличивается на 50%]: $\mu_{DR1}/2$ . |
| $\mu_{DW1}$ | 20 Гбайт· 50 Nodes / 25000 с = = 40 Мбайт/с | Пропускная способность файловой системы СУБД на запись промежуточных таблиц без сжатия [2, п. 4.3.1, рис. 3, Nodes 10, 25, 50, 100]: $V_1 \cdot Nodes/t_{bottom}$ |
| l | 20 Гбайт / $155 \cdot 10^6$ = = 130 байт | Длина записи создающей таблицы: $V_1/Q_1$. |
| Q | 134000 | Число записей в создающей таблице (R1): $p_1 \cdot Q_1$. |
| V | 134000·130 байт = = 17.4 Мбайт | Объем создающей таблицы: Q· l. |
| g | 134000 | Число хеш-разделов создающей таблицы: g=Q (оптимальное значение). |
| $l_{зонд}$ | 1 Гбайт / $18 \cdot 10^6$ = = 56 байт | Длина записи зондирующей таблицы: $V_2/Q_2$. |
| $Q_{зонд}$ | $18 \cdot 10^6$ | Число записей в зондирующей таблице (R2): $p_2 \cdot Q_2$. |
| $V_{зонд}$ | $18 \cdot 10^6 \cdot 56$ байт = = 1 Гбайт | Объем зондирующей таблицы: $Q_{зонд} \cdot l_{зонд}$ |
| v | 64 Кбайт | Размер блока ввода/вывода на диск. |
| **MapReduce (Hadoop)** | | |
| L | 2 | На каждом узле запускались два экземпляра Map и один экземпляр Reduce [2, п. 4.1.1]. |
| $\mu_{DR1}$ | (20 Гбайт+1 Гбайт) / 600 с =35 Мбайт/с | Пропускная способность файловой системы базы данных NoSQL (HDFS) на чтение: $(V_1+V_2)/T_{M11}$ (см. табл.1). |
| $\mu_{DW1}$ | 35/(22.3/5.2)=8 Мбайт/с | Пропускная способность файловой системы базы данных NoSQL (HDFS) на запись: $\mu_{DR1}$/(Average IO rate read /Average IO rate write) [16]. |
| $t_Z$ | 10 с | Время распространения функций Map и Reduce по узлам системы [2, п. 5.1.2]. |
| **Пропускные способности устройств** | | |
| $\mu_{PW}$, $\mu_{N1}$, $\mu_{N2}$,$\mu_{PR}$ | 1 Гбит/с, 128 Гбит/с, 64 Гбит/с, 1 Гбит/с | Пропускные способности порта коммутатора (выход), коммутирующей матрицы коммутатора, соединительного кольца, порта коммутатора (вход) [2, п. 4.1.2 ]. |
| СУБД: $\mu_{DR2}$, MR: $\mu_{DR2}$, $\mu_{DR3}$, $\mu_{DR4}$ | 50 Мбайт/с | Производительность диска SATA1 на последовательное чтение [12]. |
| СУБД: $\mu_{DW2}$, MR: $\mu_{DW2}$, $\mu_{DW3}$, $\mu_{DW4}$ | 50 Мбайт/с | Производительность диска SATA1 на последовательную запись [12]. |

Обозначения параметров в первой колонке табл. 1 и 2 соответствуют обозначениям, принятым в [9, 10].

В результате калибровки модели были получены следующие значения процессорного времени одной короткой логической операции алгоритма (КЛОА): СУБД – $\tau = 2 \cdot 10^{-8}$ с, MapReduce – $\tau = 1{,}6 \cdot 10^{-8}$ с.

На рис. 1 ÷ 4 приведены результаты вычислительных экспериментов, выполненных по формулам (26) [9], (38) [10], для СУБД и для технологии MapReduce (MR).

В отличие от натурных экспериментов [2, п. 4.3.4] при моделировании учитывалось, что в параллельной строчной СУБД соединяемые таблицы фрагментированы по своим первичным ключам. Эта предпосылка является более правдоподобной.

Далее в модельных экспериментах для MR увеличивалось число узлов n = = 100÷1900 с шагом 200. При этом объемы обрабатываемых таблиц UserVisits и Rankings в каждом узле оставались неизменными (см. параметры $V_1$, $Q_1$, $V_2$, $Q_2$ в табл. 2).

Так как параллельные СУБД устанавливаются, в основном, в системах с числом узлов меньше 100 [4], то при моделировании СУБД число узлов оставалось неизменным n = 100. Но при этом изменялись объемы обрабатываемых таблиц UserVisits и Rankings в каждом узле: $mV_1$, $mQ_1$, $mV_2$, $mQ_2$, m = 1÷19.

Как и ожидалось, при низкой селективности атрибута UV.visitDate (селективность = диапазон в запросе / весь доступный диапазон) и ограничении на число отсортированных результирующих записей (Y = 1) параллельная СУБД показывает существенно меньшее время выполнения запроса (рис. 1а). Основная доля времени для MR приходится на выполнение функции Map 1-й фазы (это подтверждают и натурные эксперименты). Но при n > 4500 (m > 45) MapReduce показывает лучший результат, чем СУБД (на графике это не показано).

При низкой селективности атрибута UV.visitDate и отсутствии ограничения на число отсортированных результирующих записей (Y=n/l – no limit, т.е. без ограничения) технология MR начинает выигрывать по времени при n >1500 (m >15) (рис. 1б).

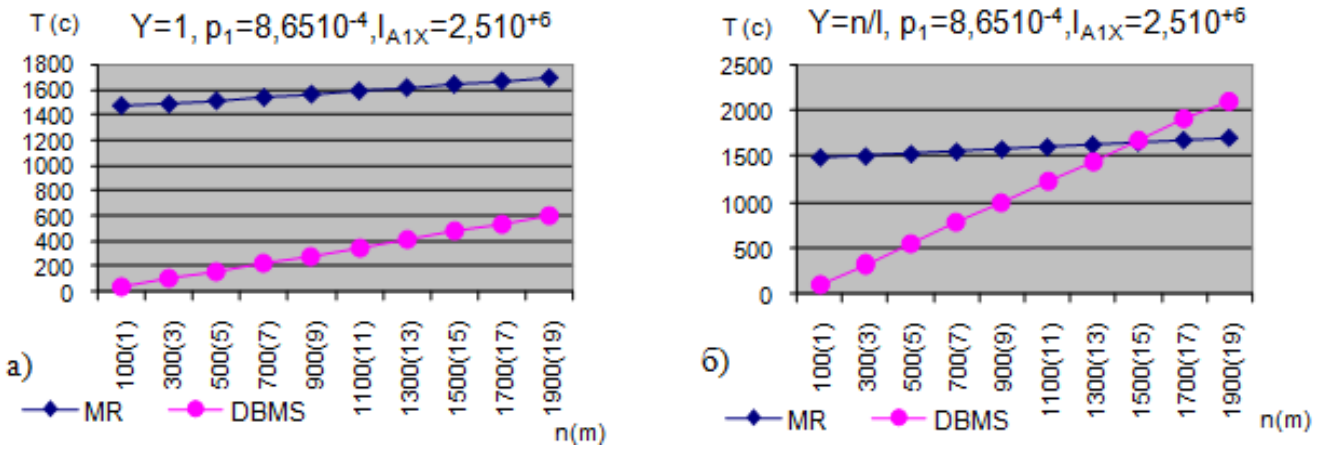


*Рис. 1.* Зависимости времени выполнения запроса при низкой селективности ($p_1$) атрибута UV.visitDate, при Y = 1 (а) и при отсутствии параметра LIMIT (б).

Это объясняется тем, что на 4-й фазе СУБД должна выполнить сортировку/объединение большого числа записей, поступающих со всех n = 100 узлов ($T_{F4}$ = 1500 с при m = 19). В то же время для MR записи, подлежащие агрегированию и сортировке, распределяются по большому числу узлов и требуемые операции выполняются параллельно (поэтому $T_{M3}+T_{R3}$= 25,6 с при n = 1900).

При средней селективности ($p_1$) атрибута UV.visitDate технология MR начинает выигрывать у СУБД уже при n > 900 (m > 9) для Y = 1 и при n > 100 (m > 1) для Y = n/l (рис. 2 а,б). Это объясняется резким увеличением времени хеш-соединения таблиц в СУБД.

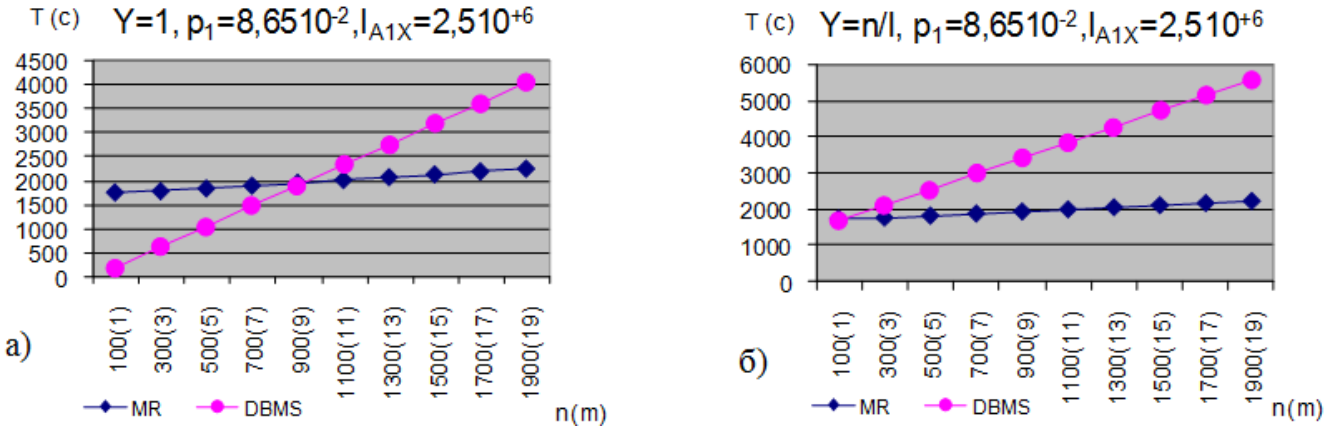


*Рис. 2.* Зависимости времени выполнения запроса при средней селективности ($p_1$) атрибута UV.visitDate, при Y = 1 (а) и при отсутствии параметра LIMIT (б).

Если уровень селективности атрибута UV.visitDate высок ($p_1$ = 1), т.е. когда индексы не используются и из базы данных читаются все записи соединяемых таблиц, то при n > 300 (m > 3) выигрыш при использовании MapReduce становится ощутимым (рис. 3 а,б). Для СУБД резко возрастает время хеш-соединения таблиц и межмашинного обмена, это связано с тем, что записи таблицы UserVisits не фрагментированы по атрибуту соединения ($T_{F1}$ = 9500 с, $T_{F2}$ = 25341 с при Y = 1 и m = 19).

Для MR возрастает время чтения и разбора записей, время пересылки записей по сети и их соединения/сортировки, а также время агрегирования записей в узлах ($T_{M1}$ = 2560 с, $T_{R1}$=4090 с., $T_{M2}$ = 625 с, $T_{R2}$ =1770 с при Y = 1 и n = 1900).

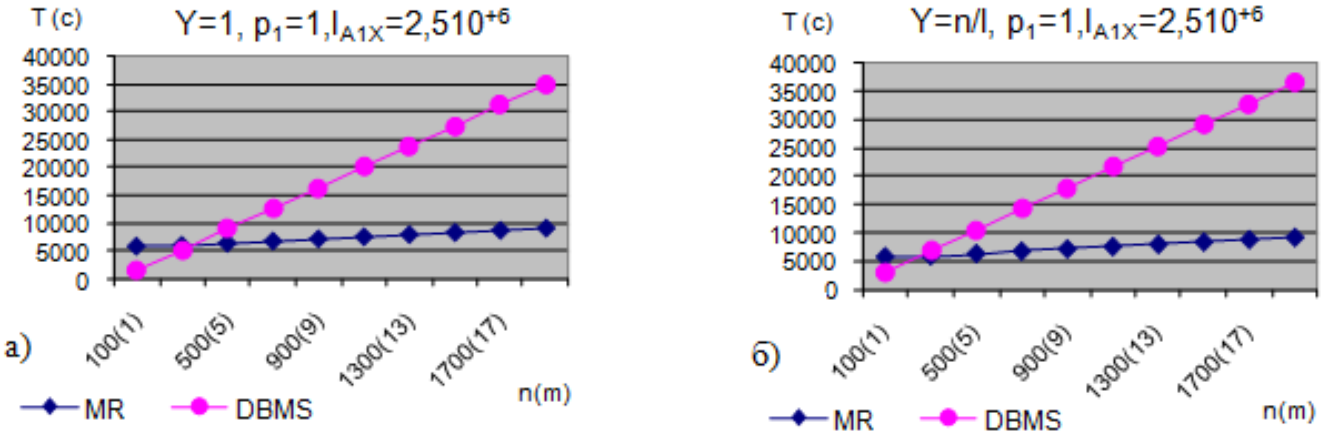


*Рис. 3.* Зависимости времени выполнения запроса при высокой селективности ($p_1$ = 1) атрибута UV.visitDate, при Y = 1 (а) и при отсутствии параметра LIMIT (б).

На рис. 4а,б показаны графики зависимостей времени выполнения запроса в случае пропорционального увеличения мощности атрибута группирования UV.sourceIP: IA1X = m·2,5·10 + 6; m = 1÷19. При средней селективности атрибута UV.visitDate и Y = 1 технология MR предпочтительнее при n > 800 (m > 8) (рис. 4а). При отсутствии ограничения на число отсортированных результирующих записей (Y = n/l) MR начинает выигрывать у СУБД уже при n > 100 (m > 1) (рис. 4б). Скачок времени выполнения запроса в СУБД при увеличении уровня объема данных ‘m’ в узле с 7 до 9 (см. рис. 4б) объясняется резким скачком времени сортировки сгруппированных записей в узле на 3-й фазе (TF3 возросло от 80 с до 9190 с).

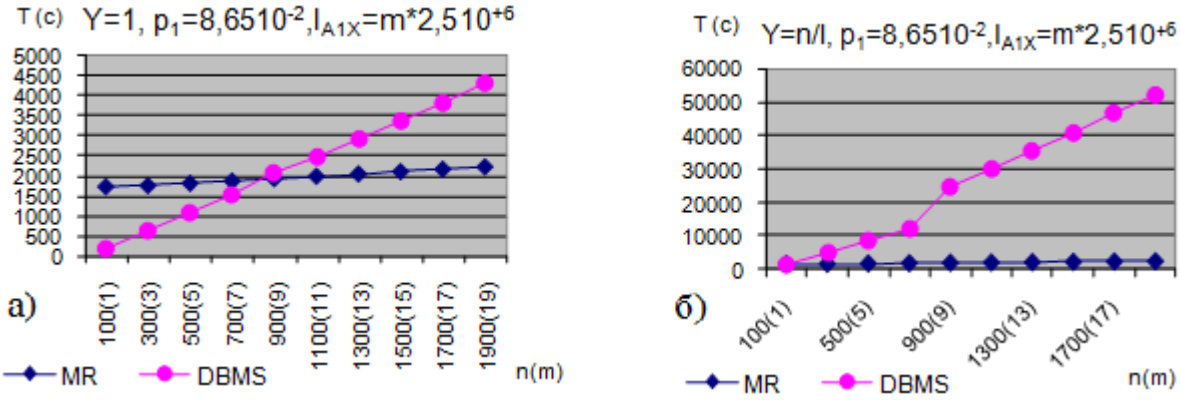


*Рис. 4.* Зависимости времени выполнения запроса при средней селективности ($p_1$) атрибута UV.visitDate при пропорциональном увеличении мощности $I_{A1X}$ атрибута группирования UV.sourceIP (m), при Y = 1 (а) и при отсутствии параметра LIMIT (б).

## Заключение

Выполнено сравнение среднего времени соединения таблиц в строчной параллельной СУБД и по технологии MapReduce (MR). Здесь объем данных увеличивался следующим образом: для MR – за счет наращивания узлов (n = 100÷1900), для СУБД – за счет наращивания объема данных в каждом узле (n = 100, m = 1÷19). При сравнительно небольших объемах обрабатываемых данных СУБД имеет преимущество. Но при увеличении объемов данных СУБД теряет преимущество и технология MapReduce начинает выигрывать по производительности (см. рис. 1÷4). Это ставит под сомнение мнение некоторых авторов [2], что MR-системы (на примере Hadoop) не могут конкурировать с параллельными СУБД по производительности при обработке структурированных данных. В конце концов, масшабируемость побеждает!

Появились гибридные СУБД, сочетающие преимущества параллельных СУБД и MR-систем. В дальнейшем предполагается разработать адекватные математические модели для систем SQL/MR [5] и MR/SQL [4] и выполнить на них сравнительные эксперименты.

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 22.01.2014.

2. *Павло Э., Паулсон Э., Разин А., Абади Д. и др.* Сравнение подходов к крупномасштабному анализу данных. Пересказ Сергея Кузнецова. [Электронный ресурс] [http://citforum.ru/database/articles/mr_vs_dbms/] Проверено 22.01.2014.
3. *Stonebraker M., Abadi D., Dawitt D.J., Madden S., Paulson E., Pavlo A., Rasin A.* MapReduce and Parallel DBMSs: Friends or Foes? Communications of the ACM, January 2010. – Vol. 53, No. 1. – P.64-71.
4. *Abouzeid A., Bajda-Pawlikowski K., Abadi D., Silberschatz A., Rasin A.* HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. In Proceedings of the Conference on Very Large Databases, August 24-28, 2009, Lyon, France.
5. *Friedman E., Pawlowski P., Cieslewicz J.* SQL/MapReduce: A practical approach to self-describing, polymorphic, and parallelizable userdefined functions. In Proceedings of the Conference on Very Large Databases, August 24-28, 2009, Lyon, France.
6. *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.
7. *Григорьев Ю.А., Плужников В.Л.* Оценка времени соединения таблиц в параллельной системе баз данных // Информатика и системы управления. – 2011. – № 1. – С.3-16.
8. *Григорьев Ю.А., Ермаков Е.Ю.* Оценка времени соединения двух таблиц в параллельной колоночной системе баз данных // Вестник МГТУ им. Н.Э. Баумана. – 2012. – № 4. – Сер. "Приборостроение": – С.80-100.
9. *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.
10. *Григорьев Ю.А., Плутенко А.Д.* Оценка времени соединения таблиц в базе данных NoSQL по технологии Mapreduce // Информатика и системы управления. – 2014. – № 1. – С.3-14.
11. *Michael G. Noll*. Benchmarking and Stress Testing an Hadoop Cluster With TeraSort, TestDFSIO & Co. [Электронный ресурс] [http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench/] Проверено 22.01.2014.
12. Скорость жесткого диска (IDE, SATA1,2,3). [Электронный ресурс] [http://mstreem.ru/skorost-zhyostkogo-diska-ide-sata123/] Проверено 22.01.2014.
13. *Григорьев Ю.А., Плутенко А.Д.* Теоретические основы анализа процессов доступа к распределенным базам данных. – Новосибирск: Наука, 2002.
14. *Григорьев Ю.А., Плутенко А.Д.* Теория и практика проектирования систем на основе баз данных: Учебное пособие. – Благовещенск: Амурский гос. ун-т, 2007.
15. *Grigoriev Y.A., Burdakov A.V., Ploutenko, A.D.* On the Estimation of Query Execution Time in Object-Oriented Databases at the Early Design Stages. Advances in Databases and Information Systems: Materials of 6ts East-European Conference, ADBIS 2002, Bratislava, Slovakia, September 8-11, 2002 (Research Communications). – Vol. 2. – P.52-64.
16. *Graefe G.* Implementing Sorting in Database Systems. ACM Computing Surveys. – 2006. – Vol. 38, No. 3. – P.1-37.
17. *Lewis J.* Cost-Based Oracle Fundamentals. Apress, 2005.

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