УДК 004.657

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

АНАЛИЗ РЕАЛИЗАЦИИ МИНИМАЛЬНОГО АЛГОРИТМА В КЛАСТЕРНОЙ АРХИТЕКТУРЕ

Рассмотрена теорема о минимальном алгоритме применительно к задаче сортировки кортежей в компьютерном кластере. Приведено более подробное ее доказательство, в котором устранены выявленные неточности. Рассмотрен распространенный алгоритм сортировки TeraSort в системе MapReduce, состоящий из двух заданий. Получены верхние оценки времени реализации первого задания в системе Hadoopна каждой фазы: Map, Shuffle, Reduce. Показано, что при фиксированной вероятности фильтрации кортежей нарушаются свойства минимального алгоритма.
Ключевые слова: минимальный алгоритм, сортировка, MapReduce, Hadoop.

Введение

В настоящее время начинают разрабатываться теоретические основы кластерных систем, построенных с использованиембаз данных NoSQL и каркаса MapReduce. Соответствующие работы вызывают повышенный интерес у научного сообщества. Так, на публикацию [1], посвященную реализации минимальных алгоритмов в системе MapReduce, имеется более десятка ссылок.

В данной статье на примере задачи сортировки записей выполнен анализ возможности реализации свойств минимального алгоритма в распространенной кластерной среде Hadoop [2].

Сначала дадим определение минимального алгоритма [1].

Обозначим через S множество входных объектов, которые используются в процессе решения задачи (проблемы). Пусть n – это мощность S (число объектов в S) и t – число машин (узлов), используемых в системе. Определим $m = n/t$, а именно m – это число объектов в одной машине, когда S равномерно распределяется по всем машинам. Рассмотрим алгоритм решения задачи на множестве объектов S. Мы говорим, что алгоритм является минимальным, если он обладает следующими свойствами.

1. *Минимальный след*: в любой момент времени каждая машина использует только O(m) пространство хранения. По определению $f = O(m)$ означает, что $\exists C(m > m_0 \Rightarrow f < C \cdot m)$, где C – некоторая константа.

2. *Ограниченный net-трафик*: в каждом задании (Job) каждая машина посы-

лает и принимает самое большое O(m) слов информации по сети.

3. *Ограниченное число заданий*: алгоритм должен завершить выполнение за фиксированное число заданий.

4. *Оптимальное вычисление*: каждая машина выполняет только O(T_{SEQ}/t) объем вычислений в общей сложности (т.е. суммируя по всем заданиям), где T_{SEQ} – это время, необходимое для решения той же задачи на одной последовательной машине. Иначе говоря, алгоритм должен достичь ускорения t, используя t параллельно работающих машин.

Алгоритм сортировки TeraSort

Пусть задан набор S из n объектов из упорядоченного домена. Обозначим через $M_1,\ldots, M_t$ машины в системе MapReduce. Первоначально S распределяется по этим машинам, каждая из которых хранит O(m) объектов, где $m = n/t$. В конце сортировки все объекты, переданные в M_i, должны быть отсортированы в этом узле и предшествовать объектам из M_j для любых $1 \le i < j \le t$.

Алгоритм сортировки TeraSort:

Задание 1(Job1):

Фаза Map.

Каждый локальный объект из M_i ($1 \le i \le t$) с вероятностью ρ посылается на один узел M_1.

Фаза Reduce (только для M_1).

Пусть S_{samp} – множество объектов (образцов), полученных узлом M_1 от других узлов, и $s = |S_{samp}|$.

Узел M_1 сортирует множество S_{samp} и выбирает *граничные объекты* b_i ($1 \le i \le t - 1$), где b_i – это $i \cdot \lceil s/t \rceil$-й объект в отсортированном множестве S_{samp}.

Примечание: перед выполнением функции Reduces неизвестно.

Задание 2 (Job2):

Граничные объекты $b_1,\ldots, b_{t-1}$ пересылаются всем машинам.

Фаза Map.

Каждый узел M_i читает объект r_i из локальной памяти. Если $b_{j-1} < r_i \le b_j$ ($1 \le j \le t$, $b_0 = -\infty$, $b_t = +\infty$), то на стадии shuffle узел M_i посылает объект r_i на узел M_j.

Фаза Reduce.

Каждый узел M_i сортирует объекты, полученные им после выполнения фазы Map.

Теорема о минимальности алгоритма сортировки TeraSort

Определим $S_i = S \cap (b_{i-1}, b_i]$ для $1 \le i \le t$. Во втором задании объекты S_i поступают на узел M_i, где выполняется их окончательная сортировка. Чтобы алгоритм TeraSort был минимальным, должны соблюдаться два условия (см. свойство «ограниченный net-трафик»)

P1: $s = O(m)$, где $s = |S_{samp}|$ (см. выше), (1)

P2: $|S_i| = O(m)$ для всех $1 \le i \le t$.

Эти условия не являются тривиальными, и их требуется доказать.

Теорема 1. Если $m \geq t \cdot \ln(n \cdot t)$ и вероятность $\rho = (1/m) \cdot \ln(n \cdot t)$ (см. фазу Map задания 1), то условия P1 и P2 выполняются одновременно, с вероятностью не менее $1 - O(1/n)$.

В [1] приведено доказательство этой теоремы, которое является достаточно кратким и имеет некоторые неточности в оценке вероятности $1 - O(1/n)$. Ниже приведено более подробное доказательство, и в котором устранены выявленные неточности.

Доказательство.

Сначала приведем выражения для границ Чернова [1]. Пусть $X_1,..., X_K$ – независимые случайные величины, распределенные по закону Бернулли: $\Pr[X_i = 1] = p_i$ для $1 \leq i \leq K$. Рассмотрим случайную величину $X = \sum_{i=1}^{K} X_i$, $\mu = E[X] = \sum_{i=1}^{K} p_i$. Имеют место следующие неравенства:

$$\Pr[X \leq (1-\alpha)\mu] \leq \exp(-\alpha^2\mu/3), \Pr[X \geq (1+\alpha)\mu] \leq \exp(-\alpha^2\mu/3),$$
$$0 < \alpha < 1, \qquad (2)$$
$$\Pr[X \geq 6\mu] \leq 2^{-6\mu}.$$

Будем полагать, что $t \geq 9$.

В противном случае $m = \Omega(n)$, т.е. $\exists C(n > n_0 \Rightarrow C \cdot n < m)$. Учитывая, что $m = n/t$, это утверждение имеет место при $n_0 = 0$ и $C = 1/9$. Таким образом, (1) следует из того факта, что $s < C_1 n$, $|S_i| < C_2 n$ и $n < m/C$.

Из правила формирования S_{samp} (фаза Map задания 1) и условия теоремы следует, что $E[s] = m\rho t = t \cdot \ln(n \cdot t)$. Тогда для $\underline{\alpha} = 0.6$ и $t \geq 9$ из (2) следует

$$\Pr[s \geq 1.6 \cdot t \cdot \ln(n \cdot t)] \leq \exp(-0.12 \cdot t \cdot \ln(n \cdot t)) \leq 1/n. \qquad (3)$$

Таким образом, свойство P1 в (1) нарушается с вероятностью, не превышающей $1/n$, так как по условию теоремы $m \geq t \cdot \ln(n \cdot t)$. Теперь проанализируем свойство P2 при условии, что

$$s < 1.6 \cdot t \cdot \ln(nt) \leq 1.6m, \qquad (4)$$

т.е. $s = O(m)$ – см. условие P1.

Предположим, что объекты S отсортированы в порядке возрастания. Разделим отсортированный список на $\lfloor t/8 \rfloor$ подсписков, по возможности равномерно. Назовем каждый подсписок отрезком (bucket). Каждый отрезок имеет от $8n/t = 8m$ до $16m$ объектов. Поясним правую границу (т.е. $16m$):

$n/\lfloor t/8 \rfloor = n/k = m/(k/(8k + i)) \leq m/(1/(8 + 7)) < 16m$, так как $i < 8$.

Если каждый отрезок покрывает по крайней мере один интервал $(b_{i-1}, b_i]$, то имеет место свойство P2 (i). Действительно, в этом случае каждый интервал $(b_{i-1}, b_i]$ пересекает не более двух отрезков, т.е. $|S_i| \leq 16m + 16m = 32m$, $|S_i| = O(m)$ для всех $1 \leq i \leq t$. Докажем (i).

Некоторый отрезок β включает интервал $(b_{i-1}, b_i]$, если β покрывает более чем $1.6 \cdot \ln(nt)$ объектов из множества S_{samp}. Учитывая неравенство (4), имеем $1.6 \cdot \ln(nt) > s/t$ (интервал включает $\lceil s/t \rceil$ последовательных объектов-примеров, полученных после выполнения фазы Map задания 1), что и доказывает (i). Найдем теперь вероятность этого события для всех отрезков (ii).

Пусть $|\beta| \geq 8m$ – число объектов в β. Определим случайную величину x_j, $1 \leq j \leq |\beta|$, $x_j = 1$, если j-й объект отрезка является объектом-примером из S_{samp}

(с вероятностью ρ из условия теоремы) и $x_j = 0$ – в противном случае. Определим

$$X = \sum_{j=1}^{|\beta|} x_j = |\beta \cap S_{samp}|. \quad (5)$$

Ясно, что $E[X] \geq 8m\ \rho = 8 \cdot \ln(nt)$. Получим

$$\Pr[X \leq 1.6 \cdot \ln(nt)] = \Pr[X \leq (1 - 4/5)8 \cdot \ln(nt)] \leq \Pr[X \leq (1 - 4/5)E[X]]$$
$$(\text{из } 2) \leq \exp(-(16/25) \cdot (E[X]/3)) \leq \exp(-(16/25) \cdot (8 \cdot \ln(nt)/3)) \leq \quad (6)$$
$$\leq \exp(-\ln(nt)) \leq 1/(nt).$$

Введем обозначение: $\overline{A_i}$ – событие, что для i-го отрезка выполняется неравенство $X_i \leq 1.6 \leq \ln(nt)$. Тогда

$$\Pr[\bigcup_i \overline{A_i}] \leq \sum \Pr[X_i \leq 1.6 \cdot \ln(nt)] \leq (t/8) \cdot (1/(nt)) = 1/(8n). \quad (7)$$

Пусть B – событие, что для всех отрезков выполняются неравенства $X_i > 1.6 \cdot \ln(nt)$. Ясно, что $B = \overline{\bigcup_i \overline{A_i}} = \bigcap_i A_i$. Из (7) с учетом (4) получим

$$\Pr[P2|P1] \geq \Pr[B] \geq 1 - 1/(8n). \quad (8)$$

Первое неравенство в (8) объясняется тем, что в качестве благоприятных событий могут выступать и события $\overline{A_i}$ для которых $X_i \geq s/t$. Оценка вероятности (*ii*) получена.

С учетом (8) и (3) имеем

$$\Pr[P2|P1] \cdot \Pr[P1] = \Pr[P2 \cap P1] \geq (1 - 1/(8n)) \cdot (1 - 1/n) =$$
$$= 1 - 9/(8n) + 1/(8n^2) \geq 1 - 9/(8n) = 1 - O(1/n). \quad (9)$$

В [1] приведена более грубая оценка $1 - O(1/n) = 1 - 17/(8n)$.

Теорема доказана.

На основании этой теоремы в [1] делается вывод об оптимальности алгоритма сортировки TeraSort:

1. *Минимальный след* – по условию.

2. *Ограниченный net-трафик* – см. условия (1) и доказанную теорему 1.

3. *Ограниченное число заданий* равное 2 (см. алгоритм сортировки TeraSort).

4. *Оптимальное вычисление* − на фазе Reduce задания 2 выполняется сортировка $O(m)$ объектов (см. свойство P2). Стоимость этой операции равна $O(m \cdot \log m) = O((n \cdot \log n)/t)$.

Таким образом, кажется, что алгоритм позволяет достичь ускорения t, используя t параллельно работающих машин (см. последнее свойство 4). Но автор [1] допускает ошибку в анализе 4-го свойства оптимального алгоритма, он в своих выкладках не учитывает реальных процессов, выполняющихся в системе MapReduce.

В статье пока рассматривается процесс реализации 1-го задания алгоритма сортировки TeraSort с учетом описания работы Hadoop [3,4].

Анализ процесса сортировки TeraSort в системе Hadoop (Задание 1)

На рис. 1 приведена уточненная схема функционирования MapReduce(MR) при выполнении какого-либо задания [3].

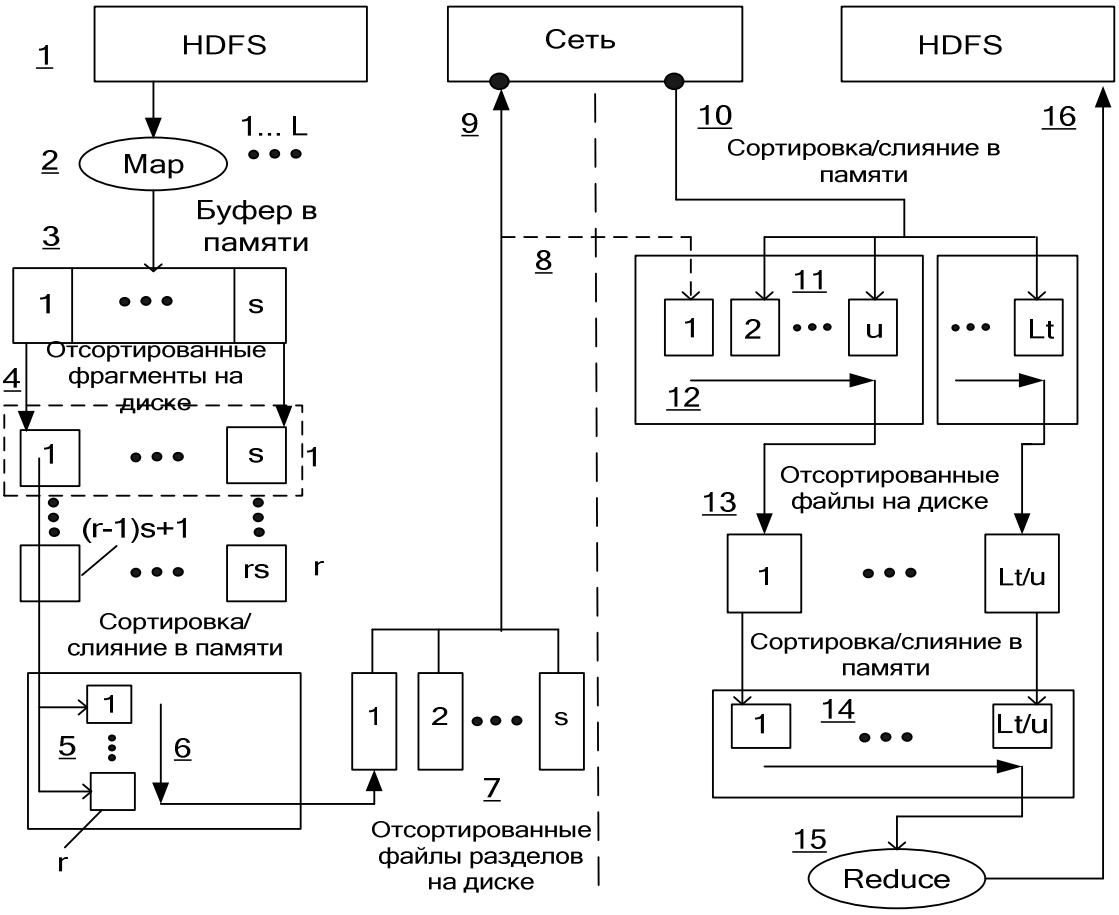

Рис. 1. Схема сортировки записей по технологии MapReduce.

Фаза Map. На каждом узле M_i запускаются L экземпляров функции Map (метка 2, метки на рис. 1 подчеркнуты), которые выполняются параллельно и читают из файловой системы MR записи (объекты) таблицы R1 в виде пары <ключ, значение> (метка 1). С вероятностью ρ запись принимается, с вероятностью $(1 - \rho)$ – отсеивается.

Число записей, принятых на обработку одной функцией Map, равно v_1. Из (2) получим

$$\Pr[v_1 \geq (1 + \alpha)\mu_{v1}] \leq \exp(-\alpha^2\mu_{v1}/3) = q_{v1},\ 0 < \alpha < 1, \quad (10)$$

$$\mu_{v1} = \mathrm{E}[v_1] = (m/L)\cdot\rho, \quad (11)$$

т.е.

$$v_1 < C_{v1}\cdot\mathrm{E}[v_1],\ C_{v1}=(1 + \alpha),\ v_1 = \mathrm{O}(\mu_{v1}) \quad (12)$$

с вероятностью

$$p_{v1} \geq 1 - q_{v1}. \quad (13)$$

Принятые записи преобразуются в записи <атрибут, пусто> и выводятся в буфер памяти (метка 3), где «атрибут» − это атрибут, по которому выполняется сортировка.

Каждой функции Map выделяется буфер памяти объемом Q_B (по умолчанию он равен 100 Мбайт [5]). Когда объем заполнения буфера превышает определенный порог P_T(по умолчанию он равен 80%), то запускается отдельная задача, которая выполняется в фоновом режиме. Она выделяет в буфере фрагменты и сортирует каждый фрагмент по ключу (in-memory sort). Число фрагментов в бу-

фере 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 (spill to disk) (метка 4). Таким образом, общее число отсортированных фрагментов, которые будут сохранены на диске в процессе выполнения всех функций Map узла, будет равно $r \cdot s \cdot L = r \cdot L$ (так как $s = 1$), где $r \cdot L$ – число сохраненных файлов, r рассчитывается по формуле (по описанию в [4]):

$$r = Q/(Q_B \cdot P_T \cdot P_M/16) = Q/Q_M, \quad (14)$$

где Q – общее число выходных записей, сформированных одним экземпляром функции Map; Q_B – размер буфера памяти (100Мбайт); P_T – порог заполнения буфера (0.8); P_M – доля буфера, отведенного под метаданные (0.05); L – число экземпляров функции Map, запущенных на одном узле.

Примечания.

1. Фрагменты $(j-1)s + 1,\ldots, js$ – это части j-го файла, $j = 1,\ldots,$ r (рис. 1, метка 4), т.е. на диске сохраняются r физических файлов, каждый из которых содержит s логических файлов (фрагментов).

2. 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) [5]. Для некоторых задач это позволяет уменьшить объем обрабатываемых данных. Но для рассматриваемого случая сортировки записей такая возможность должна быть отключена.

После того, как функция Map обработает записи, в узле запускается специальная процедура. Она сортирует записи *i*-го раздела и объединяет их в один файл (merge on disk). 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 «атрибут» (*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 каком-либо блоке ОП исчерпаны, то в этот блок подгружаются записи из связанного с ним файла. После обработки таким способом *r* фрагментов будет сформирован отсортированный по ключу «атрибут» файл *i*-го раздела(метка 7).Описанная выше процедура последовательно выполняется для всех разделов $i = 1,\ldots, s$ экземпляра Map.

В данном задании число разделов *s* равно 1, поскольку число функций Reduce равно 1 (все новые записи пересылаются на один узел, – например, M_1).

Описанные выше действия выполняются параллельно для всех экземпляров Map, запущенных на данном узле. Таким образом, будет сформировано L файлов для каждого раздела.

Время чтения и обработки записей таблицы одной функцией Марузла и их

сортировки/объединения можно представить в виде следующего выражения:

$$T_{M1} = t_Z + \max\{\frac{1}{\mu_{DR1}}\sum_{i=1}^{m/L} a_{1i},\ t_{PR12} + \frac{1}{\mu_{DW2}}(\xi = \sum_{i=1}^{v_1} a_{2i})\} + \\ + \max\{\frac{1}{\mu_{DR2}}\xi + t_{PR13},\ \frac{1}{\mu_{DW2}}\xi\}. \quad (15)$$

Чтение записей и их обработка функцией Map выполняются последовательно (метки 1 и 2 на рис. 1), а обработка записей в буфере и запись их в файлы выполняются в фоновом режиме, т.е. параллельно (метки 3 и 4) – первый max в формуле. Чтение записей фрагментов и их обработка (сортировка) осуществляются также последовательно (метки 5 и 6), а их запись в результирующий выходной файл – параллельно (метка 7) – второй max в формуле.

Все величины в равенстве (15) являются случайными (кроме m и L). Поясним обозначения в формуле (15): t_Z – время распространения функций Map и Reduce по узлам системы; a_{1i} – длина исходной записи <ключ, значение>; a_{2i} – длина выходной записи <атрибут, пусто>; μ_{DR1} – пропускная способность файловой системы MR на чтение (HDFS Hadoop); μ_{DW2} – пропускная способность локального диска на запись; μ_{DR2} – пропускная способность локального диска на чтение; t_{PR12} – процессорное время сортировки записей в буфере памяти (sort); t_{PR13} – процессорное время сортировки/слияния записей разделов; v_1 – число записей, принятых на обработку одной функцией Map, выражение (12) дает верхнюю оценку этой случайной величины,

В [3] приведены оценки средних значений процессорных составляющих на основе числа коротких логических операций алгоритма (КЛОА). Здесь получены оценки соответствующих случайных величин сверху.

А. $t_{PR12} = 6.5\tau\sum_{i=1}^{r} d_i \log_2 d_i$ – **метка 3**, (16)

где τ – время выполнения одной КЛОА; r – количество буферов памяти, которые помещены на диск, в процессе выполнения функции Map, при обработке записей таблицы R1, оно определяется выражением (14) и равно

$$r = \lceil v_1/Q_M \rceil, \quad (17)$$

здесь d_i –число записей в очередном i-м буфере, которые сортируются при заполнении буфера (так как $s = 1$):

$$d_i = \max j_i : \sum_{u=k_{i-1}+1}^{k_i = k_{i-1}+j_i} 1 \le Q_M, \mathrm{k}_0 = 0, \mathrm{k}_\mathrm{i} \le v_1. \quad (18)$$

Оценим (16) сверху. Используя (17) и (12), получим

$$r < \lceil C_{v1}\mu_{v1}/Q_M \rceil = \lceil C_{v1}\cdot m\cdot\rho/L/Q_M \rceil = r^{\max}, \quad (19)$$

с вероятностью p_{v1} (13).

1. Для $i < r^{\max}$:

$$d_i = \lceil Q_M \rceil. \quad (20)$$

2. Для $i = r^{\max}$:

$$d_i < v_1 - (r^{\max}-1)\lceil Q_M \rceil \le \lceil C_{v1}\mu_{v1} \rceil - (r^{\max}-1)\lceil Q_M \rceil = d^*, \quad (21)$$

с вероятностью p_{v1} (13).

Подставляя (19), (20) и (21) в (16), получим

$$t_{PR12} < 6.5\tau^{\max}((r^{\max}-1)\lceil Q_M \rceil \log_2 \lceil Q_M \rceil + d^* \log_2 d^*), \quad (22)$$

с вероятностью p_{v1} (13).

Б. $t_{PR13} = \tau \cdot v_1 \cdot (2.5(r-1)+4)$ – **метка 6.** (23)

Оценим (23) сверху. Из (12) и (19) получим

$$t_{PR13} < \tau^{\max} \cdot \lceil C_{v1}\mu_{v1} \rceil \cdot (2.5(r^{\max}-1)+4), \quad (24)$$

с вероятностью p_{v1} (12).

Получим теперь оценку времени T_{M1}.Подставляя (24), (22), (12) в (15), имеем

$$T_{M1} < t_Z^{\max} + \max\{\frac{m \cdot a_1^{\max}}{L \cdot \mu_{DR1}^{\min}},$$

$$6.5\tau^{\max}((r^{\max}-1)\lceil Q_M \rceil \log_2 \lceil Q_M \rceil + d^* \log_2 d^*) + \frac{\lceil C_{v1} \cdot m \cdot \rho / L \rceil \cdot a_2^{\max}}{\mu_{DW2}^{\min}}\} + \quad (25)$$

$$+\lceil C_{v1} \cdot m \cdot \rho / L \rceil \cdot \max\{\frac{a_2^{\max}}{\mu_{DR2}^{\min}} + \tau^{\max} \cdot (2.5(r^{\max}-1)+4), \frac{a_2^{\max}}{\mu_{DW2}^{\min}}\},$$

с вероятностью p_{v1} (13), где $r^{\max}$, d^*, C_{v1} определяются выражениями (19), (21) и (12). Из (25) можно получить следующую оценку:

$$T_{M1} < (1/\mu_{DR1}^{\min})O(m = n/t) + (1/\mu_{DR2}^{\min})O(mp) + (\tau^{\max}/Q_M) \cdot O((m\rho)^2). \quad (26)$$

С учетом порядка значений

$$\mu_{DR}\ (\sim 10^7),\ n\ (\sim 10^{12}),\ t\ (\sim 10^3),\ \tau\ (\sim 10^{-8}),\ Q_M\ (\sim 10^5),\ \rho << 1, \quad (27)$$

из (26) видно, что первое слагаемое является определяющим, и это пока соответствует 4-му свойству минимального алгоритма (см. введение).

Фаза Shuffle. После завершения выполнения всех функций Map система MR дает команду, узел открывает файлы, подготовленные Map (см. метку 7 на рис. 1), читает их и передает записи узлу, где выполняется функция Reduce (метки 9, 10).

В [6] была получена оценка среднего времени задержки в сети на этапе Shuffle при выполнении соединения двух таблиц. По аналогии можно получить такую же оценку для рассматриваемого задания 1:

$$T_{RNW} = \max\{\frac{(\xi_j = \sum_{i=1}^{v_{1j}+\ldots+v_{Lj}} a_{2i})}{\mu_{DR2}}, \frac{\xi_j}{\mu_{PW}},$$

$$\frac{1}{\mu_{N1}}\sum_{j=2}^{t/k}\xi_j,\ \frac{1}{\mu_{N2}}\sum_{j\notin[1,t/k]}^{t}\xi_j,\ \frac{1}{\mu_{PR}}\sum_{j=2}^{t}\xi_j,\ \frac{1}{\mu_{DW2}}(\eta = \sum_{j=1}^{t}\xi_j)\}. \quad (28)$$

Все величины в равенстве (28) являются случайными (кроме t, k – число коммутаторов и L). Здесь ξ_j – объем всех файлов, читаемых с диска j-го узла и пересылаемых в узел с номером 1 (т.е. в узел M_1), где выполняется одна функция Reduce задания 1. Верхнюю оценку $v = v_{1j} + \ldots + v_{Lj}$ можно получить по аналогии с

(10) – (13):

$$\Pr[v \geq (1+\alpha)\mu_v] \leq \exp(-\alpha^2\mu_v/3) = q_v,\ 0 < \alpha < 1 \text{ (см. (2))}, \quad (29)$$

где

$$\mu_v = \mathrm{E}[v] = L\cdot\mathrm{E}[v_1] = m\cdot\rho, \quad (30)$$

$$v < C_v\cdot\mathrm{E}[v],\ C_v = (1+\alpha),\ v = \mathrm{O}(\mu_v),\ \xi_j < C_v\cdot m\cdot\rho\cdot a_2^{\max}, \quad (31)$$

с вероятностью

$$p_v \geq 1 - q_v. \quad (32)$$

Неравенства (31) выполняются для всех узлов $j = 1,\ldots, t$, с вероятностью

$$p_{vt0} \geq 1 - t\cdot q_v. \quad (33)$$

Выражение (33) следует из следующих рассуждений.

Пусть $\Pr[\overline{A}_i] \leq p_i$, $\overline{A}_i$ – отрицание некоторого события (см., например, (29)). Тогда $\Pr[\bigcup_i \overline{A}_i] \leq \sum_i p_i$ и, следовательно:

$$p = \Pr[\overline{\bigcup_i \overline{A}_i} = \bigcap_i A_i] \geq 1 - \sum_i p_i\ . \quad (34)$$

Выражение (34) будем использовать в дальнейшем при вероятностной оценке выполнения нескольких неравенств одновременно. Неравенство (34) справедливо независимо от того, совместимы и/или зависимы события A_i.

Формула (28) определяет время передачи данных через самый загруженный ресурс. Все узлы одинаковые и работают параллельно. Поясним формулу (28).

1. Файлы, сформированные функциями Марузла (их число равно L), читаются с диска (производительность чтения равна μ_{DR2}).

2. Данные поступают в порт коммутатора (выход, μ_{PW}).

3. Узлы, подключенные к тому же коммутатору, что и узел 1, передают ему данные посредством коммутирующей матрицы коммутатора (μ_{N1}, κ – число коммутаторов, соединенных в кольцо). Объем данных можно оценить по формуле:

$$\sum_{j=2}^{t/k} \xi_j < (t/k - 1)\cdot C_v\cdot m\cdot\rho\cdot a_2^{\max}, \quad (35)$$

с вероятностью

$$p_{vt1} \geq 1 - q_{vt1}, \quad (36)$$

где $q_{vt1} = \exp(-\alpha^2(t/k - 1)m\rho/3)$.

Формулы (35), (36) выводятся по аналогии с (29) – (32) – схема Бернулли (см. (2)).

4. Остальные узлы передают данные узлу 1 через соединительное кольцо (μ_{N2}).

$$\sum_{j\notin[1,t/k]}^{t} \xi_j < (t - t/k)\cdot C_v\cdot m\cdot\rho\cdot a_2^{\max}, \quad (37)$$

с вероятностью

$$p_{vt2} \geq 1 - q_{vt2}, \quad (38)$$

где $q_{vt2} = \exp(-\alpha^2(t - t/k)m\rho/3)$.

5. Данные со всех узлов поступают в порт коммутатора одного узла 1 (вход, μ_{PR}).

$$\sum_{j=2}^{t} \xi_{\mathrm{j}} < (\mathrm{t}-1) \cdot C_{\mathrm{v}} \cdot m \cdot \rho \cdot \mathrm{a}_2^{\max}, \tag{39}$$

с вероятностью

$$p_{vt3} \geq 1 - \mathrm{q}_{\mathrm{vt3}}, \tag{40}$$

где $q_{vt3} = \exp(-\alpha^2(t-1)m\rho/3)$.

6. Файлы, сформированные функциями Map всех узлов кластера (их число равно $t \cdot L$), сохраняются на диске узла 1 (производительность записи равна μ_{DW2}).

$$\eta = \sum_{j=1}^{t} \xi_{\mathrm{j}} < \mathrm{t} \cdot C_{\mathrm{v}} \cdot m \cdot \rho \cdot \mathrm{a}_2^{\max}, \tag{41}$$

с вероятностью

$$p_{vt4} \geq 1 - q_{vt4}, \tag{42}$$

где $q_{vt4} = \exp(-\alpha^2 tm\rho/3)$.

Подставляя (31), (35), (37), (39), (41) в (28) и учитывая оценки вероятностей (33), (36), (38), (40), (42), а также выражение (34), получим

$$T_{RNW} < C_{\mathrm{v}} \cdot m \cdot \rho \cdot \mathrm{a}_2^{\max} \max\{\frac{1}{\mu_{DR2}^{\min}}, \frac{1}{\mu_{PW}^{\min}}, (t/k-1)\frac{1}{\mu_{N1}^{\min}}, (t-t/k)\frac{1}{\mu_{N2}^{\min}}, (t-1)\frac{1}{\mu_{PR}^{\min}}, t\frac{1}{\mu_{DW2}^{\min}}\}, \tag{43}$$

с вероятностью

$$p_{vt} \geq 1 - t \cdot q_v - q_{vt1} - q_{vt2} - q_{vt3} - q_{vt4} \approx 1 - t \cdot q_v. \tag{44}$$

Учитывая порядок интенсивностей

$$\mu_{PW}\ (\sim 10^8),\ \mu_{N1}\ (\sim 10^{10}),\ \mu_{N2}\ (\sim 10^9),\ \mu_{PR}\ (\sim 10^8), \tag{45}$$

а также (27), имеем

$$T_{RNW} < (1/\mu_{DW2}^{\min})\mathrm{O}(m \cdot \rho \cdot t = n \cdot \rho), \tag{46}$$

т.е. при фиксированной вероятности ρ

$$T_{RNW} = \mathrm{O}(n) \text{ и } s = \mathrm{O}(n), \tag{47}$$

что свидетельствует о невыполнении свойства 2 (ограниченный net-трафик) минимального алгоритма (см. также (1)).

Подставляя в (46) выражение для ρ из теоремы 1 и учитывая ограничение, накладываемое на m в этой теореме, получим

$$T_{RNW} = \mathrm{O}(t \cdot \ln(n \cdot t)) = \mathrm{O}(m) \text{ и } s = \mathrm{O}(m), \tag{48}$$

что подтверждает вывод теоремы о выполнении свойства 2 минимального алгоритма для этого случая.

Фаза Reduce. Узел 1 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 функции Reduce, система

группирует входные записи по значению атрибута, по которому выполняется сортировка (метка 15). Далее функция Reduce выбирает граничные объекты $b_i (1 \le i \le t-1)$ и выводит их в файловую систему HDFS (метка 16). Здесь b_i – это $i \cdot \lceil mp \rceil$-й объект в отсортированном множестве S_{samp}.

Время сортировки/объединения записей и их обработки на фазе Reduce можно представить в виде следующего выражения:

$$\mathrm{T_{R1}} = \max\{\frac{\eta}{\mu_{DR2}} + \tau \cdot (\chi = \sum_{i=1}^{tL/u} \sum_{j=1}^{u} v_{ij}) \cdot (2.5(u-1)+4), \frac{\eta}{\mu_{DW2}}\} + \\ + \frac{\eta}{\mu_{DR2}} + \tau \cdot \chi \cdot (2.5(tL/u-1)+4) + \frac{\sum_{i=1}^{t-1} \mathrm{a_{2i}}}{\mu_{DW1}}. \quad (49)$$

Поясним формулу (49).

1. С диска узла 1 читаются файлы, сформированные функциями Map всех узлов кластера, η – это объем всех файлов (см. последнюю сумму в (28)), число файлов равно $t \cdot L$. Производительность чтения равна μ_{DR2}. Верхняя оценка η равна (41) с вероятностью (42).

2. Файлы объединяются в группы по «u» файлов в каждой, число групп равно $t \cdot L/u$. Выполняется сортировка/слияние файлов в каждой группе по аналогии с (23). По аналогии с (41)

$$\chi < t \cdot C_v \cdot m \cdot \rho, \quad (50)$$

с вероятностью (42).

3. $t \cdot L/u$ файлов общим объемом η сохраняются на диске со скоростью μ_{DW2}. Запись выполняется в фоновом режиме.

4. Далее эти файлы считываются с диска, производительность чтения равна μ_{DR2}.

5. Выполняется сортировка/слияние файлов в оперативной памяти и записи передаются на вход функции Reduce без предварительного запоминания.

6. $(t-1)$ записей (b_i) передаются функцией Reduce в выходной поток и сохраняются в файловой системе HDFS Hadoop.

Подставляя верхние оценки составляющих в (49), получим

$$\mathrm{T_{R1}} < \mathrm{t} \cdot C_{\mathrm{v}} \cdot m \cdot \rho \cdot [\max\{\frac{\mathrm{a_2^{max}}}{\mu_{DR2}^{\min}} + \tau^{\max} \cdot (2.5(u-1)+4), \frac{\mathrm{a_2^{max}}}{\mu_{DW2}^{\min}}\} + \\ + \frac{\mathrm{a_2^{max}}}{\mu_{DR2}^{\min}} + \tau^{\max} \cdot (2.5(tL/u-1)+4)] + \frac{(t-1)\mathrm{a_2^{max}}}{\mu_{DW1}^{\min}}, \quad (51)$$

с вероятностью (42). Учитывая, что

$$t \le 4000 \text{ (для Hadoop)}, \; L \le 2 \text{ (как правило)}, \; u \sim 10^2, \quad (52)$$

перепишем (51) следующим образом:

$$\mathrm{T_{R1}} < \mathrm{t} \cdot C_{\mathrm{v}} \cdot m \cdot \rho \cdot (\frac{\mathrm{O(a_2^{max})}}{\mu_{DR2}^{\min}} + \tau^{\max} \cdot O(u)). \quad (53)$$

При фиксированном ρ

$$T_{R1} = \mathrm{O}(n), \quad (54)$$

что свидетельствует о невыполнении свойства 4 (оптимальное вычисление) минимального алгоритма (см. также (1)).

Подставляя в (53) выражение для ρ из теоремы 1 и учитывая ограничение, накладываемое на m в этой теореме, получим

$$T_{R1} = O(t \cdot m \cdot \ln(n \cdot t)/m) = O(m), \quad (55)$$

что подтверждает вывод теоремы о выполнении свойства 4 минимального алгоритма для этого случая. Но вероятность $\rho = (1/m) \cdot \ln(n \cdot t)$ из теоремы 1 имеет порядок $\sim 10^{-8}$, что вряд ли можно реализовать на практике с достаточной точностью.

Следует отметить, что при $a_2^{\max} \sim 10$ и в силу ограничений (27) и (52) составляющая ввода/вывода и процессорная составляющая в (53) имеют одинаковый порядок 10^{-6} (два слагаемых в скобках). Это свидетельствует о необходимости учитывать в расчетах процессорное время.

Заключение

Приведено подробное доказательство теоремы о минимальности распространенного алгоритма сортировки TeraSort (теорема 1) при его реализации в кластерной системе MapReduce, в котором устранены выявленные неточности, допущенные в [1].

Рассмотрен процесс выполнения первого задания (Job1) алгоритма сортировки TeraSort в системе MapReduce на примере Hadoop. Получены верхние границы времени реализации этого задания на каждой фазе: Map, Shuffle, Reduce, а также вероятности надежности таких оценок. На основании этих оценок показано, что при фиксированной вероятности ρ фильтрации сортируемых записей на фазе Map задания 1 нарушаются свойства 2 и 4 минимальности алгоритма. Правда, они сохраняются, если для расчета ρ ($\sim 10^{-8}$) использовать выражение из теоремы 1.

В следующей публикации на эту тему планируется рассмотреть второе задание (Job2) выполнения сортировки записей по технологии MapReduce и проверить выполнение условий минимальности алгоритма.

ЛИТЕРАТУРА

1. *Tao Y., Lin W., Xiao X.* Minimal mapreduce algorithms // In SIGMOD Conference. – 2013. – P. 529-540,
2. HadoopProject: [Электронный ресурс]. [http://hadoop.apache.org]. Проверено 23.03.2015.
3. *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.
4. *Palla Konstantina*. A Comparative Analysis of Join Algorithms Using the Hadoop Map/Reduce Framework // Master of Science School of Informatics University of Edinburgh. – 2009. – P.1-93.
5. Hadoop, HDFS, MapReduce and Hive – Some salient understandings: [Электронный ресурс]. [http://hadoop-gyan.blogspot.ru/]. Проверено 23.03.2015.
6. *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.

E-mail:
Григорьев Юрий Александрович – grigorev@bmstu.ru.