О параллельном доступе к базе данных при ограниченном количестве обращений к внешним носителям информации

М.Н. Назаров

Рассмотрим задачу параллельного извлечения информации из базы данных по r независимым адресам при допуске ограниченного количества обращений к внешним носителям. При этом будем использовать модель, в которой база данных представлена в виде таблицы для булевой функции n переменных. Случай, допускающий одно единственное обращение, был успешно рассмотрен в [3]. При этом под процессором в указанной работе понимается булева схема постоянного размера, которая может выполнять запросы к внешней памяти, затрачивая определенное время, обозначенное **extime**. Представленный в [3] алгоритм использует для своей работы очень большое количество таких процессоров. Здесь мы исследуем, как повлияет на оценки сложности алгоритма из [3] допуск нескольких обращений к внешним носителям при сохранении основной структуры алгоритма. Отметим, что дальнейшее развитие методы из [3] получили при решении задачи распределения данных по модулям памяти с целью предотвращения конфликтов доступа к каждому модулю [1, 2].

Предположим, что при решении задачи параллельного доступа допускается t обращений к внешним носителям. Ясно, что t должно быть строго меньше количества запросов r, так как иначе задача становится тривиальной и не требует применения каких-либо специальных алгоритмов. Действительно, если t больше r, то достаточно одного процессора, который последовательно считает все необходимые r бит за r обращений.

Определим взаимосвязь величины t и оценок сложности алгоритмов доступа. Рассмотрим сначала оценки алгоритма $DP_{r,f}$ [3] при новых условиях, а затем изменим первоначальный алгоритм для минимизации количества процессоров в оценках сложности. При этом мы будем непосредственно использовать описание и оценки процедур из [3]. Более подробное описание первоначального алгоритма и его процедур можно найти в [3] и [4].

Отметим, что количество процессоров в первоначальном алгоритме определялось из условия максимальной распараллеленности всех процедур и существенно превышает количество процессоров, необходимых для обращения к внешним носителям. Ясно, что допуск нескольких обращений не повлияет на количество процессоров, требующихся для всех процедур. Время работы алгоритма увеличится только на время $t-1$ обращения к внешнему носителю. Единственное преимущество увеличения допустимого числа обращений к внешним носителям в первоначальном алгоритме заключается в уменьшении необходимого количества самих носителей. Действительно, если, например, допустить два обращения на одном шаге, то на одном носителе можно объединить информацию, ранее хранимую на двух. Следовательно, в общем случае при t обращениях количество носителей можно уменьшить в t раз.

Рассмотрим теперь влияние новых условий на оценки сложности при отказе от максимальной распараллеленности алгоритма. Определим минимальное количество процессоров, необходимое для обеспечения одного обращения к внешним носителям при сохранении методов распределения базы данных по носителям из [2, 3]. А затем допустим t обращений и оценим сложность полученного алгоритма.

Заметим, что для работы алгоритма $DP_{r,f}$ требуется представить булеву функцию f через булевы функции f_A и $g_l : \{0,1\}^{n-4k} \to \{0,1\}$, где $k = \lceil 3\log r + \log n \rceil$. Для большей наглядности разбиение функции f можно представить в виде таблицы для булевых функций от $n-4k$ переменных.

Здесь $m = 2^{4k}$, а $p = 2^{4k} \cdot \frac{1}{n}$. Допуск одного обращения означал, что для любых r входов из каждого столбца можно было извлечь только один бит.

Отметим, что на третьем шаге алгоритма $DP_{r,f}$ [3] для всех $i = 1, 2, ..., r$ вычисляется

y_1	y_2	...	y_{n-4k}	f_{A1}	f_{A2}	...	f_{Am}	g_1	g_2	...	g_p
0	0	...	0	*	*	...	*	*	*	...	*
0	0	...	1	*	*	...	*	*	*	...	*
...	...	...	...	...	...	...	...	...	...	...	...
1	1	...	1	*	*	...	*	*	*	...	*

Таблица 1.

$$\mathbf{B}_6(A_i, u_i, \vec{Z}, \vec{Y}) = Z_{l(A_i,u_i)} \oplus \left(\bigoplus_{A \in l^{\#}(A_i,u_i)} Y_A \right) =$$

$$= g_{l(A_i,u_i)}(B_i) \oplus \left(\bigoplus_{A \in l^{\#}(A_i,u_i)} f_A(B_i) \right) = f(A_i, B_i).$$

Из последней формулы легко видеть, что для каждой строки, соответствующей набору B_i, требуется считать только одно значение из столбца для функции $g_{l(A_i,u_i)}$ и по одному значению из каждого столбца для функций f_A, где $A \in l^{\#}(A_i, u_i)$. Таким образом, из каждой строки таблицы 1 необходимо считать $|l^{\#}(A_i, u_i)| + 1 = 2^k$ значений. Для всех r строк эти значения не пересекаются, так как согласно главному ограничению алгоритма допускалось считывание только одного значения из каждого столбца. Следовательно, всего необходимо считать $r2^k$ значений и именно столько должно быть процессоров, чтобы обеспечить одно обращение к внешним носителям. Причем $k = \lceil 3 \log r + \log n \rceil$. Итак, мы показали, что минимальное количество процессоров, необходимое для соблюдения главного ограничения алгоритма оценивается величиной

$$r2^k \leqslant 2r^4 n.$$

Однако при таком количестве процессоров необходимо изменить алгоритм $DP_{r,f}$, так как его шаги, начиная со второго, нас уже не удовлетворяют.

Будем считать, что для каждой из r строк, соответствующих входным наборам B_1, B_2, ... , B_r, у нас сопоставлено 2^k процессоров. Пусть для каждой из указанных строк $\vec{P}_i$ представляет собой

набор длиной 2^k из переменных P_A, где $A \in GF(q)^4$. Тогда получаем следующий новый алгоритм.

Алгоритм $MDP_{r,f}$:

1. Применим алгоритм $\boldsymbol{A}$ ко входу $(A_1, A_2, ..., A_r)$, то есть $\vec{u} = \boldsymbol{A}(\vec{A})$.

2. Для всех $i = 1, 2, ... , r$ параллельно выполним:

2.1. Применим процедуру, определяющую наборы $\vec{P}_i$:

а) для всех $A \in GF(q)^4$ в $\vec{P}_i$ запомним A такие, что $\boldsymbol{B}_1(A_i, u_i, A) = 1$;

б) для всех $l \in SL_4(U)$ вычислим $\boldsymbol{B}_2(A_i, u_i, l)$ и положим $Q_i = l$: $\boldsymbol{B}_2(A_i, u_i, l) = 1$.

2.2. Вычислим выходы булевых функций базы данных, то есть

а) для всех $A \in \vec{P}_i$ вычислим $Y_{A,i} = f_A(B_i)$;

б) для $l = Q_i$ вычислим $Z_i = g_l(B_i)$.

Заметим, что это единственный шаг, на котором происходит обращение к внешним носителям. На этом шаге считываются $r2^k$ значений из базы данных, причем не более одного значения из каждой части, хранимой на отдельном носителе.

2.3. Применим процедуру для вычисления всех выходов алгоритма:

$$f(A_i, B_i) = Z_i \oplus \left(\bigoplus_{A \in \vec{P}_i} Y_{A,i} \right) = g_{l(A_i,u_i)}(B_i) \oplus \left(\bigoplus_{A \in l^{\#}(A_i,u_i)} f_A(B_i) \right).$$

Рассмотрим оценку времени работы этого алгоритма при допуске t обращений к внешним носителям, что позволяет уменьшить количество процессоров в t раз.

Теорема 1. *Для любых положительных целых n, r и t, $t < r$, $r \leqslant 2^{\frac{n}{16}}$ и для любых булевых функций f от n переменных время работы алгоритма $MDP_{r,f}$ при количестве процессоров, ограниченном величиной $O\left(\frac{r^4 n}{t}\right)$, можно оценить как:*

$$\mathbf{ptime}(MDP_{r,f}) = O(t \cdot r^9 n^3) + t \cdot \mathbf{extime}.$$

Доказательство. Рассмотрим первый шаг алгоритма $MDP_{r,f}$. Время его работы при указанном количестве процессорах несложно оценить, используя оценки из [3] как $\mathrm{O}(r^5 \log(rk))$.

Для второго шага учтем, что для каждого $i = 1, 2, ..., r$, то есть для каждой строки, соответствующей наборам B_i у нас имеется $\frac{2^k}{t}$ процессоров и для всех i алгоритм работает параллельно.

На шаге 2.1. необходимо $|GF(q)^4|$ раз выполнить процедуру $\boldsymbol{B}_1$ и $|SL_4(U)|$ раз выполнить процедуру $\boldsymbol{B}_2$ при помощи $\frac{2^k}{t}$ процессоров. Нам известно, что

$$\begin{gathered}\mathbf{bp}(\boldsymbol{B}_1) = \mathrm{O}(k^2),\\ \mathbf{ptime}(\boldsymbol{B}_1) = \mathrm{O}(\log k),\\ \mathbf{bp}(\boldsymbol{B}_2) = \mathrm{O}(k),\\ \mathbf{ptime}(\boldsymbol{B}_2) = \mathrm{O}(\log k).\end{gathered}$$

Следовательно, $\frac{2^k}{t\cdot k^2}$ процедур $\boldsymbol{B}_1$ может быть выполнено за время $\mathbf{ptime}(\boldsymbol{B}_1)$, а $|GF(q)^4|$ процедур $\boldsymbol{B}_1$ будет выполнено за время

$$\frac{t\cdot k^2\cdot \left|GF(q)^4\right|}{2^k}\cdot \mathbf{ptime}(B_1).$$

Аналогично, получаем, что $|SL_4(U)|$ процедур $\boldsymbol{B}_2$ будет выполнено за время

$$\frac{t\cdot k\cdot |SL_4(U)|}{2^k}\cdot \mathbf{ptime}(B_2).$$

Таким образом, время выполнения шага 2.1. оценивается как

$$\mathrm{O}(\log k)\cdot\left(\frac{t\cdot k^2\cdot\left|GF(q)^4\right|}{2^k}+\frac{t\cdot k\cdot|SL_4(U)|}{2^k}\right)\leqslant \mathrm{O}(\log k)\cdot\left(\frac{t\cdot k^2\cdot 2^{4k}}{2^k}+\frac{t\cdot k\cdot 2^{4k}}{2^k n}\right)=$$
$$=\mathrm{O}(\log k)\cdot t\cdot k2^{3k}\cdot\left(k+\tfrac{1}{n}\right)=\mathrm{O}(t\cdot k^2 2^{3k}\log k).$$

На шаге 2.2. происходит t обращений к базе данных. Время одного обращения оценивается как **extime**, поэтому время выполнения этого шага оценим как $t\cdot\mathbf{extime}$.

Рассмотрим теперь заключительный шаг алгоритма $MDP_{r,f}$, на котором при помощи $\frac{2^k}{t}$ параллельных процессоров требуется сложить 2^k значения булевых переменных. Построив схему из функциональных элементов, легко видеть, что время выполнения этого шага можно оценить как

$$\mathrm{O}(t + \log 2^k) = \mathrm{O}(k + t).$$

Окончательно получаем, что время работы всего алгоритма при ограниченном количестве процессоров определяется как

$$\mathbf{ptime}(MDP_{r,f}) = \mathrm{O}(r^5 \log(rk)) + \mathrm{O}(t \cdot k^2 2^{3k} \log k) + \mathrm{O}(k+t) + t \cdot \mathbf{extime}.$$

Учитывая, что $k = \lceil 3 \log r + \log n \rceil$, получаем

$$\mathbf{ptime}(MDP_{r,f}) = \mathrm{O}(t \cdot r^9 n^3) + t \cdot \mathbf{extime}.$$

Что и требовалось доказать.

Очевидно, что размер базы данных не изменится при увеличении допустимого количества обращений к базе данных, если не изменять структуру хранения данных. Отметим только, что допуск t обращений к носителям информации может обеспечить снижение в t раз необходимого количества носителей.

Список литературы

[1] Andreev A.E., Clementi A.E.F., Penna P., Rolim J.D.P. Memory Organization Schemes for Large Shared Data: A Randomized Solution for Distributed Memory Machines // 16th Annual Symposium on Theoretical Aspects of Computer Science STACS'99. 1999. LNCS 1563.

[2] Andreev A.E., Clementi A.E.F., Penna P., Rolim J.D.P. Parallel Read Operations Without Memory Contention // Technical Report TR00-053 of the ECCC. 2000.

[3] Andreev A.E., Clementi A.E.F., Rolim J.D.P. On the Parallel Computations of Boolean Functions on Unrelated Inputs // IV IEEE Israel Symposium on Theory of Computing and Systems (ISTCS'96). IEEE. 1996. P. 155 – 161.

[4] Назаров М.Н. О параллельном вычислении булевых функций на нескольких независимых наборах и оптимизации параллельного доступа к криптографически защищенным базам данных // Интеллектуальные системы. Т. 6. Вып. 1 – 4. 2001. С. 303 – 332.