УДК 681.3.06 : 51

ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ СОРТИРОВКИ

Г.Г.Иванов

Потребности повышения быстродействия вычислительных машин привели к построению многопроцессорных вычислительных систем (МПВС), обладающих высокой суммарной скоростью работы и большим объемом оперативной памяти. Примерами систем такого рода являются универсальные вычислительные системы [1] и система ILLIAC-IV [2].

Однако если за показатель эффективности МПВС принять время решения одной задачи, а не пакета специально подобранных задач, то оказывается, что многие из существующих алгоритмов плохо распараллеливаются и на отдельных этапах решения часть процессоров не используется. Поэтому с целью максимального использования присущего МПВС параллелизма ставится задача переосмысления процессов решения задач и поиска параллельных алгоритмов на основе системы г р у п п о в ы х операций, выполняемых МПВС, т.е. макроопераций над массивами определенной структуры [3].

С другой стороны, целесообразно, не ориентируясь на определенную структуру МПВС, провести анализ наиболее распространенных классов задач с целью выявления возможности их распараллеливания. Результатом такого анализа должна быть система групповых операций, которая дает исходную информацию для проектирования МПВС [4].

В настоящей статье предлагаются параллельные алгоритмы сортировки (упорядочения) массивов чисел, ориентированные на систему типа [5], и приводятся некоторые оценки их эффективности. Алгоритмы записываются на языке APL [6,7], позволяющем получить короткое и наглядное описание, причем параллелизм алго-

ритмов отражается самой структурой языка.

§ I. Основной алгоритм сортировки

Пусть $\underline{a}=(a_0,a_1,\ldots,a_{n-1})$ - исходный неупорядоченный массив. Под сортировкой понимается упорядочение компонент $\underline{a}$ в порядке неубывания слева направо. В терминах языка APL программа сортировки определяется следующим алгоритмом:

Алгоритм I

I. $\underline{k} \leftarrow \theta/\underline{a}$

2. $\underline{b} \leftarrow \underline{k}\,\underline{a}$

где $\underline{b}$ - результирующий упорядоченный массив.

Операции алгоритма I являются достаточно сложными для аппаратной реализации, и их нужно выразить через более простые операции APL , причем так, чтобы компоненты векторов $\underline{k}$ и $\underline{b}$ определялись параллельно.

Введем следующие операции:

I. $\underline{k} \leftarrow \tilde{\theta}/\underline{a}$, где k_i присваивается число, равное номеру a_i в упорядоченном массиве.

2. $\underline{b} \leftarrow \underline{k}\,\tilde{\underline{a}} \leftrightarrow b_i \leftarrow a_{k_i}$

Теперь сортировку можно определить алгоритмом П.

Алгоритм П

I. $\underline{k} \leftarrow \tilde{\theta}/\underline{a}$

2. $\underline{b} \leftarrow \underline{k}\,\tilde{\underline{a}}$

Полученный алгоритм П существенно проще алгоритма I, т.к. операции $\underline{b} \leftarrow \underline{k}\,\underline{a}$ и $\underline{b} \leftarrow \underline{k}\,\tilde{\underline{a}}$ по своей сложности эквивалентны, а операция $\underline{k} \leftarrow \theta/\underline{a}$ эквифинальна любой из следующих последовательностей операций:

I. $\underline{\ell} \leftarrow \tilde{\theta}/\underline{a}$	I. $\underline{\ell} \leftarrow \tilde{\theta}/\underline{a}$
2. $\underline{k} \leftarrow \underline{\ell}\,\tilde{\underline{\iota}}(n)$	2. $\underline{k} \leftarrow \underline{\ell}\,\iota\,\underline{\iota}(n)$

тогда как параллельное вычисление $\underline{k} \leftarrow \tilde{\theta}/\underline{a}$ не представляет труда и будет описано ниже.

Определим более подробно операцию $\underline{k} \leftarrow \tilde{\theta}_0/\underline{a}$ (индексация начинается с нуля). Вычисление k_i сводится к подсчету числа компонент $\underline{a}$, меньших a_i и числа компонент $\underline{a}$, стоя-

щих слева от компоненты a_i и равных ей. Вектор $\underline{k}$ определяется следующими формулами:

$$u_{ij} = \begin{cases} 1, & \text{если } a_i > a_j, \\ 0, & \text{если } a_i \leqslant a_j; \end{cases} \tag{1}$$

$$v_{ij} = \begin{cases} 1, & \text{если } a_i = a_j \text{ и } j < i, \\ 0 & \text{- в остальных случаях;} \end{cases} \tag{2}$$

где $i, j \in J$, $J = \{0, 1, \ldots, n-1\}$;

$$k_i = \sum_{j=0}^{n-1} u_{ij} + \sum_{j=0}^{n-1} v_{ij}. \tag{3}$$

ПРЕДЛОЖЕНИЕ 1. Все компоненты вектора $\underline{k}$, определяемого формулами (1) - (3), различны, т.е. для любых значений a_ρ и a_τ ($\rho, \tau \in J$) $k_\rho \neq k_\tau$.

ДОКАЗАТЕЛЬСТВО. В соответствии с формулой (3)

$$k_\rho = \sum_{j=0}^{n-1} u_{\rho j} + \sum_{j=0}^{n-1} v_{\rho j},$$

$$k_\tau = \sum_{j=0}^{n-1} u_{\tau j} + \sum_{j=0}^{n-1} v_{\tau j}.$$

1. Если $a_\rho = a_\tau$, тогда $\sum_{j=0}^{n-1} u_{\rho j} = \sum_{j=0}^{n-1} u_{\tau j}$, но $\sum_{j=0}^{n-1} v_{\rho j} \neq \sum_{j=0}^{n-1} v_{\tau j}$, и если $\rho < \tau$, то $\sum_{j=0}^{n-1} v_{\rho j} < \sum_{j=0}^{n-1} v_{\tau j}$.

2. Если $a_\rho > a_\tau$, то $\sum_{j=0}^{n-1} u_{\rho j} > k_\tau$, следовательно, $k_\rho > k_\tau$.

3. Если $a_\tau > a_\rho$, то $\sum_{j=0}^{n-1} u_{\tau j} > k_\rho$, следовательно, $k_\tau > k_\rho$.

СЛЕДСТВИЕ 1. Для любого $i \in J$ $k_i \in J$.

СЛЕДСТВИЕ 2. С помощью вектора $\underline{k}$ можно установить взаимно однозначное соответствие между компонентами вектора $\underline{a}$ и вектора $\underline{b}$, получаемого операцией перестановки $\underline{b} \leftarrow \underline{k}\,\tilde{\int}\,\underline{a}$, причем компоненты вектора $\underline{b}$ будут расположены в порядке неубывания.

Основным алгоритмом сортировки является алгоритм Ш.

Алгоритм Ш

1. $\underline{A} \leftarrow \underline{\varepsilon}(n) \overset{\circ}{\times} \underline{a}$
2. $\underline{B} \leftarrow \underline{\iota}^{\circ}(n) \dagger \underline{A}$
3. $\underline{U} \leftarrow (\underline{A} > \underline{B})$
4. $\underline{V} \leftarrow \square(n \times n) \times (\underline{A} = \underline{B})$
5. $\underline{k} \leftarrow +/(\underline{U} \vee \underline{V})$
6. $\underline{W} \leftarrow \underline{k} \Downarrow (\underline{\varepsilon}^{t}(n) \overset{\circ}{\times} \underline{\varepsilon}(n))$
7. $\underline{b} \leftarrow \underline{W} \overset{\circ}{\rangle} \underline{a}$

В алгоритме Ш строка 3 формирует все значения u_{ij}, строка 4 – все значения v_{ij} и строка 5 вычисляет значения всех компонент вектора $\underline{k}$. Последовательность строк I – 5 является развернутой записью операции $\underline{k} \leftarrow \tilde{\theta}_{\circ}/\underline{a}$, а строки 6 и 7 реализуют операцию $\underline{b} \leftarrow \underline{k} \tilde{\int}_{\circ} \underline{a}$.

§ 2. Сортировка больших массивов

Если матрица процессоров [5] имеет размеры $n \times n$, то, применяя алгоритм Ш, можно упорядочить массив длины n. Сортировку массива длины ρn ($\rho > 1$) можно выполнять с помощью алгоритма Ш, разбив исходный массив на ρ частей и вычисляя вектор $\underline{k}$ клеточными методами. Однако этот способ неприемлем как вследствие большого количества вычислений, необходимых для получения вектора $\underline{k}$ длины ρn, так и по причине значительных логических трудностей, связанных с перестановкой компонент исходного массива в соответствии со значением $\underline{k}$.

В этом параграфе приводятся более простые и эффективные методы сортировки больших массивов, основанные на применении алгоритма попарной сортировки (алгоритм IУ).

Пусть $\underline{d}$ – исходный неупорядоченный массив длины ρn и $\underline{d}^{t} = (d_0^{t}, d_1^{t}, \ldots, d_{n-1}^{t})$ – t-й подмассив $\underline{d}$, где $t \in T = \{1, 2, \ldots, \rho\}$. Упорядочим каждый t-й подмассив $\underline{d}$ с помощью алгоритма Ш.

Суть алгоритма IУ состоит в том, что до двум упорядоченным

подмассивам $\underline{d}^t$ и $\underline{d}^s$ $(s,t \in \mathcal{T})$ строится вектор

$$\underline{k} = (k_0^t, k_1^t, \ldots, k_{n-1}^t, k_0^s, k_1^s, \ldots, k_{n-1}^s) ,$$

с помощью которого упорядочивается массив

$$\underline{d}^{t \oplus s} = (d_0^t, d_1^t, \ldots, d_{n-1}^t, d_0^s, d_1^s, \ldots, d_{n-1}^s) = (\underline{d}^t, \underline{d}^s).$$

Обозначим $\underline{d}^t$ и $\underline{d}^s$ через $\underline{a}$ и $\underline{b}$, соответственно, а результирующий массив длины $2n$ - через $\underline{e}$.

Алгоритм IУ

I. $\underline{A} \leftarrow \underline{\varepsilon}(n) \overset{\circ}{\times} \underline{a}$

2. $\underline{B} \leftarrow \underline{\iota}^{\circ}(n) \downarrow (\underline{\varepsilon}(n) \overset{\circ}{\times} \underline{b})$

3. $\underline{F} \leftarrow (\underline{A} > \underline{B})$

4. $\underline{G} \leftarrow \underline{\iota}^{\circ}(n) \downarrow \neg \underline{F}$

5. $\underline{t} \leftarrow (+/\!/ \underline{F}) + \underline{\iota}^{\circ}(n)$

6. $\underline{s} \leftarrow (+/\!/ \underline{G}) + \underline{\iota}^{\circ}(n)$

7. $\underline{k} \leftarrow \underline{t} \oplus \underline{s}$

8. $\underline{W} \leftarrow \underline{k} \downarrow (\underline{\varepsilon}^1(2n) \overset{\circ}{\times} \underline{\varepsilon}(2n))$

9. $\underline{e} \leftarrow \underline{W} \overset{\circ}{\gamma} (\underline{a} \oplus \underline{b})$

Предварительное упорядочение двух массивов длины n и выполнение алгоритма IУ эквивалентно применению алгоритма Ш для массива длины $2n$.

Для сортировки массива при $\rho > 2$ выполняется упорядочение подмассивов длины n алгоритмом Ш, и затем к полученным подмассивам применяется алгоритм IУ. При этом порядок выбора подмассивов для попарной сортировки может определяться, например, методом Боуза-Нельсона [8].

Пусть t_1 - время выполнения алгоритма Ш, а t_2 - время выполнения алгоритма IУ. Тогда если $\rho = 2^h$, где h - целое и $h > 0$, общее время сортировки T определяется следующей формулой:

$$T = 2^h t_1 + (3^h - 2^h) t_2 . \qquad (4)$$

Полагая $t_1 = t_2 = t$, преобразуем (4) к виду:

$$T = 3^h t . \qquad (5)$$

Преимуществом метода Боуза-Нельсона является то, что он не требует для своего выполнения резерва памяти, однако время сортировки быстро растет с увеличением длины массива.

Существенно уменьшить время сортировки можно путем использования метода слияния по двум путям [9]. При этом предварительно упорядочиваются по алгоритму Ш подмассивы длины n, затем каждая пара упорядоченных подмассивов сливается в один упорядоченный [illegible] полного упорядочения всего массива.

Опишем [illegible]ния. Пусть имеются два упорядоченных подмассива $\underline{a} = (\underline{a}^1, \underline{a}^2, \ldots, \underline{a}^\gamma)$ и $\underline{b} = (\underline{b}^1, \underline{b}^2, \ldots \underline{b}^\gamma)$, где $\underline{a}^i$ и $\underline{b}^i$ - подмассивы длины n. На первом шаге по алгоритму IУ выполняется сортировка массива $\underline{c} = (\underline{a}^1, \underline{b}^1)$ и первая половина полученного массива $\underline{d}$, содержащая младшие элементы, записывается в результирующий массив. На втором шаге проводится сортировка оставшейся половины массива $\underline{d}$ и подмассива $\underline{a}^2$, если первый элемент $\underline{a}^2$ не больше первого элемента $\underline{b}^2$ ($a_0^2 \leqslant b_0^2$), в противном случае проводится сортировка оставшейся половины $\underline{d}$ и подмассива $\underline{b}^2$ и первая половина полученного упорядоченного массива записывается в результирующий массив. Алгоритм выполняется до полного исчерпания $\underline{a}$ или $\underline{b}$, после чего оставшиеся подмассивы просто переписываются в результирующий массив.

ПРЕДЛОЖЕНИЕ 2. Результатом описанного выше процесса слияния двух упорядоченных массивов $\underline{a}$ и $\underline{b}$ является упорядоченный массив.

ДОКАЗАТЕЛЬСТВО. Обозначим оставшиеся после k-го шага слияния части $\underline{a}$ и $\underline{b}$ через $\underline{a}(k) = (\underline{a}^i, \ldots, \underline{a}^\gamma)$ и $\underline{b}(k) = (\underline{b}^j, \ldots, \underline{b}^\gamma)$, причем $i + j = k + 3$. Обозначим оставшуюся половину массива $\underline{d}$ через $\underline{d}^k$ и результирующий массив - $\underline{\ell}$. Номера элементов обозначаются нижними индексами.

После выполнения k-го шага слияния $d_0^k \leqslant \ell_{kn-1}$. Пусть $a_0^i \leqslant b_0^j$, тогда на $(k+1)$-м шаге сортируется массив $\underline{c} = (\underline{d}^k, \underline{a}^i)$. Для доказательства предложения 2 надо показать, что выполняется неравенство

$$\ell_{n(k+1)-1} \leqslant b_0^j, \qquad (6)$$

где $\ell_{n(k+1)-1}$ - полученный на $(k+1)$-м шаге старший элемент массива $\underline{\ell}$.

I. Если $d^k_{n-1} \leqslant a^i_0$, то $l_{n(k+1)-1} = d^k_{n-1}$ и, соответственно, выполняется неравенство (6).

2. Рассмотрим случай, когда $d^k_{n-1} > a^i_0$. Поскольку $d^k_{n-1} \leqslant \max(a^i_0, b^j_0) \leqslant b^j_0$ и $l_{n(k+1)-1} \leqslant d^k_{n-1}$, то $l_{n(k+1)-1} \leqslant b^j_0$, причем $d^{k+1}_{n-1} \leqslant \max(a^{i+1}_0, b^j_0)$.

При длине исходного массива N и размерности матрицы процессоров $n \times n$ время выполнения сортировки при использовании метода слияния равно

$$T_c = \lceil N/n \rceil t_1 + \lceil N/n \rceil \lceil \log_2 N/n \rceil t_2 .$$

где $k = \lceil x \rceil$ - целое и $k \geqslant x > k-1$.

При использовании метода слияния достигается высокая скорость сортировки, однако требуется резерв памяти объемом $N/2$. Уменьшить резерв памяти можно путем использования списковой адресации, при этом время сортировки несколько увеличится.

Литература

I. Евреинов Э.В., Косарев Ю.Г. Однородные универсальные вычислительные системы высокой производительности. Новосибирск, "Наука", 1966.

2. Barnes G.H. et al. The ILLIAC-IV computer. -IEEE Trans. on Comp.,1968,C-17,N8,p.746-757.

3. Jones P.D.,Lincoln N.R.,Thornton J.E. Whither computer architecture ? -In:IFIP congress 71. Amsterdam,1971,booklet TA-4,p.162-168.

4. Иловайский И.В. Алгоритмический синтез структур многопроцессорных ЦВМ и систем. Настоящий сборник, стр. 16-33.

5. Thurber K.J.,Myrna J.W. System design of a cellular APL computer. -IEEE Trans. on Comp.,1970,C-19,N4,p.291-301.

6. Iverson K.E. A programming language. New York,Wiley,1962.

7. Иванов Г.Г., Иловайский И.В., Фет Я.И. Обзор языка программирования APL. - "Алгоритмы и алгоритмические языки", вып. 7 (в печати).

8. Bose R.C.,Nelson R.J. A sorting problem. -Journal of the ACM,1962,9,N2,p.282-296.

9. Глушков В.М., Гладун В.П., Лозинский Л.С., Погребинский С.Б. Обработка информационных массивов в автоматизированных системах управления. Киев, "Наукова думка", 1970.

Поступила в ред.-изд.отдел
22 февраля 1972 г.