Ein Verfahren zum Transponieren großer, sequentiell gespeicherter Matrizen

U. Schumann, Karlsruhe

Zusammenfassung: *Für Matrizen, deren Größe die Kapazität des schnellen Direktzugriffsspeichers einer Datenverarbeitungsanlage übersteigt, wird ein schneller Algorithmus zum Transponieren beschrieben, der allein sequentielle Datenträger benötigt.*

Wenn die Matrix zeilenweise gespeichert ist und NO die Länge der Zeilenvektoren ist, dann ist der Ein-/Ausgabe-Aufwand des Verfahrens der Summe der Primfaktoren der Zahl NO, vermehrt um die Anzahl der Faktoren, proportional. Für $1 \leqslant NO \leqslant 4096$ werden für dieses Verfahren optimale Werte von NO angegeben. Der Algorithmus wird mit einer „primitiven" Methode verglichen. Außerdem wird über Erfahrungen mit Systemen zur Verwaltung virtueller Speicher in diesem Zusammenhang berichtet.

Summary: *For transposing matrices with a size much bigger than the capacity of the fast direct access memory of a computer, a fast algorithm is described which requires only sequential external storage devices. If the matrix is stored by rows and NO is the number of elements in the row vector then the I/O-time of the method is proportional to the sum of the prime factors of NO plus the number of these factors. Optimum numbers NO are given for this method in the range $1 \leqslant NO \leqslant 4096$.*

The algorithm is compared with a "primitive" method. In addition to this, experience with systems providing virtual memory capabilities for this purpose is reported.

1. Problembeschreibung

In den unterschiedlichsten Anwendungsbereichen der Datenverarbeitung können sehr große, voll besetzte Matrizen auftreten und mit ihnen die Aufgabe, diese zu transponieren. Hierfür werden zunächst zwei Beispiele beschrieben.

Bei der Integration einer partiellen Differentialgleichung des Impuls- oder Wärmetransportes, die z. B. in Raum und Zeit nach dem Differenzenverfahren diskretisiert wurde und zeitlich schrittweise integriert wird, werden je Zeitschritt eine große Anzahl von Lösungswerten für alle Ortspunkte ermittelt. Diese wird man sinnvollerweise auf eine externe Speichereinheit, wie z. B. ein Magnetband, ausgeben. Nach Beendigung der zeitlichen Integration hat man auf dem externen Speicher (der im folgenden stets als Band bezeichnet wird) eine zweidimensionale Matrix A stehen, die aus NO Spalten (NO = Anzahl der Ortspunkte) und NZ Zeilen (NZ = Anzahl der Zeitschritte) besteht (Bild 1) und zeilenweise gespeichert ist (die Festlegung auf „Zeilen" und „Spalten" ist willkürlich, sie kann auch umgekehrt erfolgen). Vor einer Auswertung der Ergebnisse als Funktion der Zeit bei konstantem Ort ist die Matrix zu transponieren, d. h. so umzuordnen, daß sie spaltenweise gespeichert vorliegt.

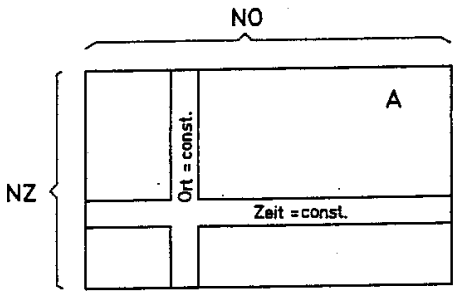

Bild 1. Aufbau der betrachteten Matrix A

Analog stellt sich die gleiche Aufgabe bei der Verarbeitung von Meßdaten eines Experimentes, die NO Meßkanälen entstammen und von einem Prozeßrechner periodisch über NZ Zyklen abgefragt und sequentiell auf ein Magnetband geschrieben werden. Bevor man nun die Meßdaten eines Kanals in einem nachfolgenden Verarbeitungsschritt z. B. einer Fourier-Analyse unterziehen kann, müssen die Daten nach Meßkanälen geordnet werden, d. h. ebenfalls, die NO*NZ-Matrix A muß transponiert werden. Dieses Problem mußte beispielsweise in SEDAP [1], einem Programmsystem zur Auswertung experimenteller Ergebnisse, gelöst werden.

In beiden Fällen handelt es sich um Matrizen, die sehr viel mehr Daten enthalten, als im schnellen Direktzugriffsspeicher (Kernspeicher) einer EDV-Anlage gespeichert werden können. Für die Integration der Differentialgleichung wird man bereits die Zahl NO der Ortspunkte so groß gewählt haben, daß bestenfalls einige wenige Lösungsvektoren für konstante Zeit im Kernspeicher untergebracht werden können. Bei der Meßdatenerfassung kann man mit Zykluszeiten von $\frac{1}{500}$ Sek. für 64 Kanäle [7] rechnen, so daß bereits nach wenigen Experimentsekunden die erzeugte Datenmenge (NO*NZ Einzeldaten) den verfügbaren Kernspeicher auch großer Rechenanlagen sprengt.

Nun könnte man daran denken, den Direktzugriffsspeicher in einfacher Weise dadurch virtuell zu vergrößern, daß man sich eines Programmsystems bedient, welches direkt zugreifbare Sekundärspeicher, wie Platten und Trommeln, als logische Erweiterung des Kernspeichers durch entsprechende Zugriffsmethoden simuliert. Derartige dynamische und virtuelle Speicherungsmethoden sind beispielsweise in den Systemen ICES [2] und MATLAN [3] implementiert. In solchen Systemen kann die gesamte Matrix A zeilenweise in den virtuellen Speicher eingelesen wer-

den. Will man jedoch anschließend die Matrix A spaltenweise ausgeben, so führt das – wie Versuche ergaben – bei den genannten Systemen zu so häufigem Umspeichern der Daten zwischen primärem und sekundärem Direktzugriffsspeicher, daß die Zeit, während der das entsprechende Programm in der EDV-Anlage verweilt, unvertretbar groß wird. Die Verweilzeit ist zu unterscheiden von der reinen Rechenzeit der Zentraleinheit des Rechners; letztere war bei den durchgeführten Versuchen um Größenordnungen kleiner als die Verweilzeit. Die Ergebnisse dieser Versuche sind um so erstaunlicher, als in MATLAN der speziell für das Transponieren vorgesehene Befehl benutzt wurde. Zudem wird man derartige Systeme zusammen mit den erforderlichen sekundären Direktzugriffsspeichern zur Simulation sehr großer Kernspeicher nicht an jeder Anlage verfügbar haben.

Betrachtet man die Aufgabe, eine zeilenweise auf ein Band geschriebene, sehr große Matrix zu transponieren, naiv, so wird man folgenden „primitiven" Algorithmus für naheliegend halten:

Zusätzlich zu dem Eingabeband, daß die zeilenweise gespeicherte Matrix enthält, wird man ein zweites Band verwenden, daß die transponierte Matrix aufnehmen soll. Sodann wird man das Eingabeband so oft durchlesen, wie die Zeilen Elemente enthalten (NO), und dabei beim ersten Durchgang aus jedem Zeilenvektor das erste Element lesen und auf das Magnetband schreiben, beim zweiten Druchgang das jeweils zweite Element usw. Hierbei muß jedoch das erste Band NO-mal durchgelesen und also entsprechend oft zurückgespult werden (in FORTRAN: "REWIND"); gleichzeitig wird das Ausgabeband einmal beschrieben und nach Beendigung der Transponierung ebenfalls zurückgespult. Insgesamt sind also (NO + 1) Lese-/Schreibdurchgänge und REWINDs erforderlich. Betrachtet man nun die wahrscheinliche Größenordnung von NO (z. B. $2^{12} = 4096$ im ersten und $2^6 = 64$ im zweiten Beispiel), so erkennt man, daß diese große Anzahl von Lese-/Schreibdurchgängen zu sehr großen Verweilzeiten führt.

Es soll nun ein Verfahren angegeben werden, bei dem, ebenfalls unter Verwendung zweier sequentieller Datenträger („Bänder"), die Anzahl der Lese-/Schreibdurchgänge und REWINDs zumindest auf die Summe der Primfaktoren, aus denen sich die Zahl NO zusammensetzt, plus der Anzahl der Primfaktoren selbst reduziert wird; im Falle von $NO = 2^{12}$ also von r = 4097 auf r = 24 + 12 = 36 bzw. von r = 65 auf r = 12 + 6 = 18. Außerdem wird gezeigt, daß der Aufwand noch dadurch verkleinert wird, daß man ein $NO_{opt} = NOPT$ mit optimaler Zerlegung ($NOPT \geqslant NO$) verwendet, und weiter dadurch, daß paarweise auftretende Primfaktoren 2 zu Zahlen 4 zusammengezogen werden. Hierdurch ergibt sich zu NO = 64 ein Wert r = 15 und zu NO = 4096 r = 30.

2. Beschreibung der vorgeschlagenen Methode

Eine Matrixtransponierung ist einer Sortieraufgabe ähnlich. Im Unterschied zum Sortieren ist das Ordnungskriterium bei der Transponierung jedoch die Position innerhalb des linearen Feldes und nicht ein Teil der Daten. Dennoch ist es nicht verwunderlich, wenn das im folgenden beschriebene Verfahren (allerdings nur für solche Werte von NO, die Potenzen von 2 sind) Ähnlichkeiten mit dem 2-Wege-Sortierverfahren [5] besitzt.

Es wird mit zwei Bändern gearbeitet; die ursprüngliche Matrix wird zerstört.

Die Methode setzt voraus, daß die Zahl NO in ihre L Primfaktoren $IP_i \geqslant 2$ mit $NO = \prod_{i=1}^{L} IP_i$ zerlegt wurde. Das ist gewissermaßen eine Umkehrung der beliebten Übungsaufgabe, die ersten NO Primfaktoren aufzufinden [4]. Sodann läuft das Verfahren in L Phasen ab: In der i-ten (i = 1, 2, . . . , L) Phase wird damit begonnen, daß die erste Information vom Eingabeband gelesen und auf das Ausgabeband geschrieben wird; anschließend wird die der zuletzt gelesenen Information nach IP_i (dem dieser Phase zugeordneten Primfaktor) Plätzen folgende Information gelesen und ebenfalls auf das Ausgabeband geschrieben usf. bis das Ende des Eingabebandes erreicht ist; anschließend wird das Eingabeband zurückgespult und bei der zweiten Information mit dem Lesen in Schritten von IP_i begonnen, dieser Vorgang wiederholt sich IP_i-mal. Sodann sind alle Daten vom Eingabeband – in veränderter Reihenfolge – auf das Ausgabeband geschrieben, beide Bänder werden zurückgespult, für die folgende Phase werden Ein- und Ausgabeband vertauscht, und die i-te Phase ist beendet. In einer Phase muß also $(IP_i + 1)$-mal zurückgespult werden. Nach L Phasen befindet sich auf dem zuletzt benutzten Ausgabeband die Transponierte der Matrix A (Beweis folgt), und es mußten bis dahin $\sum_{i=1}^{L} (IP_i + 1)$-mal die Bänder gelesen bzw. beschrieben und zurückgespult werden.

Folgendes Beispiel soll das Verfahren verdeutlichen:

Ausgangsmatrix ist eine Matrix mit NO = 6 Spalten und NZ = 2 Zeilen. Die Primfaktorzerlegung von NO ist $IP_1 = 2$, $IP_2 = 3$, L = 2. Bild 2 veranschaulicht den Lese-/Schreibprozeß.

Formal läßt sich die Vorschrift wie folgt definieren:

Gegeben sind ein Ausgangsfeld $(F_m^{(1)}, m = 1, 2, \ldots, NZ*NO)$ und eine Zerlegung $NO = \prod_{i=1}^{L} IP_i$; $IP_i \geqslant 2$ natürliche Zahl.

Das Feld $F_m^{(1)}$ enthält zeilenweise gespeichert eine Matrix A mit NO Spalten und NZ Zeilen. Nach $L \geqslant 1$ Transformationsphasen

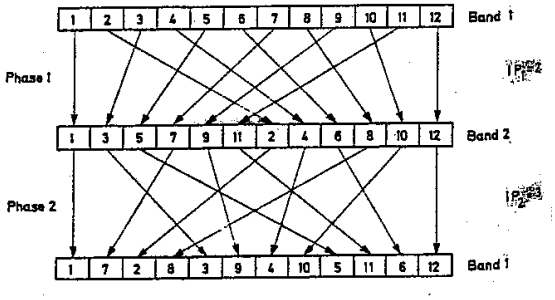

Bild 2. Beispiel für das Vefahren bei NO = 6

$F_k^{(i+1)} = F_m^{(i)}$, m = 1, 2, . . . , NZ*NO, i = 1, 2, . . . , L

enthält $F_m^{(L+1)}$ die Matrix A spaltenweise gespeichert, wenn

$k = 1 + (m-1)/IP_i + MOD\,(m-1, IP_i) * NZ*NO/IP_i$ (1)

ist; hierbei sollen für q und p ganze nichtnegative Zahlen (p > 0)

MOD (q, p) = q mod p

und q/p eine ganze Zahl sein, die die größte ganze Zahl kleiner oder gleich der rationalen Zahl q:p ist

(z. B. 7/3 = 2 und MOD (7,3) = 1).

Die Übereinstimmung der formalen Vorschrift mit der obigen Beschreibung eines praktischen Vorgehens läßt sich leicht durch Durchspielen der Vorschrift für variierendes m einsehen.

3. Beweis der transponierenden Wirkung der Vorschrift

Für den Beweis werden folgende Beziehungen verwendet:

$$\begin{aligned} &q/p = 0 \\ &MOD\,(q, p) = q \quad \text{falls } 0 \leqslant q < p;\ q, p \text{ ganz} \\ &MOD\,(q + r*p, p) = MOD\,(q, p) \quad r \text{ ganz}, r \geqslant 0 \end{aligned} \tag{2}$$

Die Position eines Matrixelementes $A_{i,j}$ einer NZ*NO-Matrix in einem linearen Feld $F^{(1)}$ ist bei zeilenweiser Speicherung:

$m = j + (i-1)*NO$ [6]. (3)

Dasselbe Element steht in der transponierten Matrix, also der spaltenweise gespeicherten Matrix A, an der Position

$k = i + (j-1)*NZ$. (4)

Ausgehend von der Transformationsvorschrift (1), läßt sich durch vollständige Induktion beweisen, daß sich das Element $A_{i,j}$ nach p > 0 Transformationen in dem linearen Feld $F^{(p+1)}$ an der Position

$$\begin{aligned} k^{(p)} = {} & 1 + \frac{J-1}{IP_1 \cdot IP_2 \cdot IP_3 \cdot \ldots \cdot IP_p} + \frac{(I-1)\cdot NO}{IP_1 \cdot IP_2 \cdot \ldots \cdot IP_p} \\ & + \left\{ MOD\,(J-1, IP_1) \cdot \frac{NO}{IP_1 \cdot IP_2 \cdot \ldots \cdot IP_p} \right. \\ & + MOD\left(\frac{J-1}{IP_1}, IP_2\right) \cdot \frac{NO}{IP_2 \cdot IP_3 \cdot \ldots \cdot IP_p} \\ & + MOD\left(\frac{J-1}{IP_1 \cdot IP_2}, IP_3\right) \cdot \frac{NO}{IP_3 \cdot IP_4 \cdot \ldots \cdot IP_p} \\ & + \ldots \\ & \left. + MOD\left(\frac{J-1}{IP_1 \cdot IP_2 \cdot \ldots \cdot IP_{p-1}}, IP_p\right) \cdot \frac{NO}{IP_p} \right\} \cdot NZ \end{aligned} \tag{5}$$

befindet. Nach L Phasen befindet sich $A_{i,j}$ also an der Position

$$\begin{aligned} k^{(L)} = {} & 1 + 0 + (I-1) \\ & + \left\{ MOD\,(J-1, IP_1) \right. \\ & + MOD\left(\frac{J-1}{IP_1}, IP_2\right) \cdot IP_1 \\ & + \ldots \\ & \left. + MOD\left(\frac{J-1}{IP_1 \cdot IP_2 \cdot \ldots \cdot IP_{L-1}}, IP_L\right) \cdot (IP_1 \cdot IP_2 \cdot \ldots \cdot IP_{L-1}) \right\} NZ \end{aligned} \tag{6}$$

Der zweite Summand auf der rechten Seite von (5) verschwindet hier, da der Nenner gleich NO ist und daher größer als der Zähler. Entsprechend wurde auch sonst NO gekürzt. Durch fortgesetztes Dividieren mit Rest kann man zeigen, daß sich jede ganze Zahl $x \geqslant 0$ darstellen läßt als

$$x = (\ldots((n_L \cdot IP_L + q_L) \cdot IP_{L-1} + q_{L-1}) \cdot IP_{L-2} + \ldots + q_2) \cdot IP_1 + q_1. \tag{7}$$

Für $x = j - 1$ muß wegen $j - 1 < NO$ und $NO = \prod_{i=1}^{L} IP_i$ $n_L = 0$ sein (für $n_L > 0$ wäre $x \geqslant NO$). Es folgt daher durch Einsetzen in (6) unter Beachtung von (2):

$k^{(L)} = I + (J-1) \cdot NZ$.

D. h. nach L Phasen befindet sich das Element $A_{i,j}$ tatsächlich an der Position, die gemäß (4) erwartet wurde.

4. Optimale Zeilenlänge NOPT

Wenn die durch das Problem vorgegebene Zahl der Elemente einer Zeile NO gerade selbst eine Primzahl ist, entspricht der hier beschriebene Algorithmus voll dem „primitiven" Verfahren. Für hinreichend große Zahlen NO ist es in solchen Fällen also sinnvoll, in einem vorbereitendem Arbeitsgang zunächst die Zeilenvektoren der Matrix mit so vielen bedeutungslosen Daten (z. B. Nullen) aufzufüllen, bis ein möglichst optimaler Wert NOPT für die Länge der Zeilenvektoren verwendet werden kann. Für die Bestimmung von NOPT ist eine sinnvolle Zielfunktion zu definieren. Da der Aufwand einerseits der Anzahl der REWIND-Operationen und andererseits der Verlängerung NOPT:NO der Zeilenvektoren proportional ist, wird

$$Z = (NOPT:NO) \cdot \left(\sum_{i=1}^{L} IP_{opt_i} + L \right) \tag{8}$$

als Zielfunktion verwendet.

Eine weitere Verbesserung läßt sich erreichen, wenn paarweise auftretende Primfaktoren 2 zu einer Zahl 4 zusammengezogen werden. Im ersten Fall wäre der Aufwand proportional 2 + 2 + 2, im zweiten nur proportional 4 + 1. Das beruht auf dem einmaligen Sonderfall, daß das Produkt zweier Primfaktoren gleich deren Summe ist.

Unter Berücksichtigung dieser Verbesserung zeigt Tabelle 1 die optimale Zahl NOPT: zu einem gegebenen NO bezeichnet diejenige Zahl NOPT das absolute Minimum der Zielfunktion (8), die die kleinste Zahl größer oder gleich NO aus der Tabelle 1 ist.

5. Schluß

Für das Transponieren sehr großer Matrizen mit NO Spalten und NZ Zeilen, die zeilenweise gespeichert vorliegen, wurde ein Verfahren vorgeschlagen, bei dem der Lese-/Schreibaufwand höchstens der Summe der Primfaktoren von NO, vermehrt um die Anzahl der Primfaktoren, proportional ist. Das Verfahren benötigt lediglich zwei sequentielle externe Datenträger. Der Direktzugriffsspeicher muß lediglich einzelne Matrixelemente (abgesehen von Blockungen) aufnehmen können.

Tabelle 1. Optimale Werte für die Länge NO der Zeilenvektoren

1	2	3	4	5
6	7	8	9	10
12	15	16	18	20
21	24	25	27	28
30	32	36	40	45
48	50	54	56	60
64	72	75	80	81
84	90	96	100	108
112	120	125	128	135
144	150	160	162	168
180	192	200	216	225
240	243	256	270	288
300	320	324	336	360
375	384	400	405	432
448	450	480	486	500
512	540	576	600	625
640	648	675	720	729
768	800	810	864	900
960	972	1024	1080	1125
1152	1200	1215	1280	1296
1344	1350	1440	1458	1500
1536	1600	1620	1728	1792
1800	1875	1920	1944	2000
2025	2048	2160	2187	2304
2400	2430	2500	2560	2592
2700	2880	2916	3072	3125
3200	3240	3375	3456	3600
3645	3840	3888	4096	

Das Verfahren bringt Vorteile gegenüber dem „primitiven", wenn die Anzahl der Daten NZ*NO so groß ist, daß nicht alle Daten im direkt zugreifbarem schnellen Speicher untergebracht werden können, und NO $\geqslant$ 8 ist. Wie Bild 3 zeigt, steigt der Aufwand für das Transponieren mit NO etwa logarithmisch an (bei NO*NZ = const.). Der beschriebene Algorithmus ist daher besonders bei großem NO vorteilhaft.

Erfahrungen bei der Verwendung der Systeme ICES [2] und MATLAN [3] für das Transponieren großer Matrizen, unter Benutzung der dort gegebenen virtuellen Speicherungskapazitäten, zeigen die Notwendigkeit auf, in derartigen Systemen schnelle Verfahren für diesen Zweck zu integrieren.

Literatur

[1] *Audoux, M.:* SEDAP – A Systematic Approach to the Processing of Experimental Data. IFIP Congress 71, Ljubljana (Aug. 1971), TA-6-71 bis 75.

[2] *Roos, D.* (ed.): ICES System: General Description. MIT, Department of Civil Engineering, Sept. 1967, R 67–49.

[3] IBM System/360 Matrix Language (MATLAN). Program Number 360A-CM-05X(H20-0564-0).

[4] *Bauer, F. L.; Goos, G.:* Informatik, 1. Teil. Springer Verlag, Berlin 1971.

[5] *Wedekind, H.:* Datenorganisation. W. de Gruyter, Berlin 1970.

[6] *Knuth:* The Art of Computer Programming, Vol. 1. Reading, Mass. (1969), S. 296.

[7] *Rittirsch, G.:* Aufbau und Betrieb der rechnergesteuerten Meßdatenerfassungsanlage im Institut für Reaktorentwicklung (in Vorbereitung).

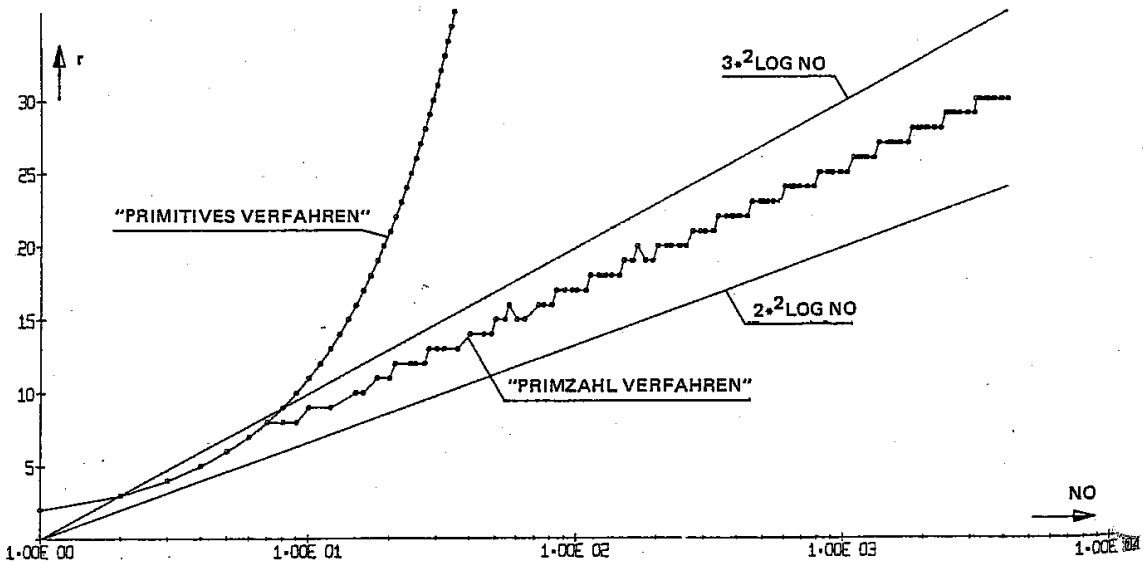

Bild 3. Ein-/Ausgabe Aufwand zum Transponieren als Funktion von NO