ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΗΜΜΥ

ΠΑΡΑΛΛΗΛΟΙ ΚΑΙ ΔΙΚΤΥΑΚΟΙ ΥΠΟΛΟΓΙΣΜΟΙ

ΣΕΤ ΑΣΚΗΣΕΩΝ 3^{Ο}

«Υλοποίηση αλγόριθμου εσωτερικού γινομένου αραιού πίνακα με διάνυσμα. Σειριακός και Παράλληλος αλγόριθμος . Χρήση COO και CRS μορφοποίησης του αραιού Πίνακα πριν την υλοποίση του εσωτερικού γινομένου.»

Αλεξανδρίδης Γιώργος 831

Γρετσίστας Βασίλης 849

Μαρέλας Γιώργος 378

ΑΣΚΗΣΗ 1η :Υλοποίηση σειριακού αλγορίθμου για το εσωτερικό γινόμενο αραιού πίνακα με διάνυσμα.(COO format for A array)

Πρωτού δείξουμε την υλοποίηση σειριακού αλγοριθμού του εσωτερικού γινομένου, παραθέτουμε τον κώδικα με τον οποίο μετατρέπουμε έναν αραιό πίνακα σε συμπιεσμένη μορφή Coordinate format. Στην συγκεκριμένη μορφοποίηση ο νέος πίνακας αποτελείται από τρία διανύσματα. Το πρώτο διάνυσμα (a_values) είναι αυτό που περιέχει όλες τις μή μηδενικές τιμές του αραιού πίνακα.Στο δεύτερο (i_index) και στο τρίτο (j_index) αποθηκεύονται αντίστοιχα ο αριθμός της γραμμής και της στήλης στην οποία βρίσκεται η τιμή που έχει αποθηκευτεί στο a_values. Τα στοιχεία του Α, αποθηκεύονται κατά γραμμές μέσα στα τρία διανύσματα. Τα στοιχεία κάθε γραμμής αποθηκεύονται με αύξουσα σειρά κατά στήλη.

Μετατροπή αραιού πίνακα σε μορφοποιήση COO :

```
for  i=0 to i = ROW-1
        for  j = 0 to j = COLLUM-1
                if   sparse[i][j] != 0
                        a_values[val] = sparse[i][j];
                        i_index[hor] = i;
                        j_index[vert] = j;
                        val++;
                        hor++;
                        vert++;
        End for j
End for i
```

$$\begin{pmatrix} 6 & 0 & 9 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 5 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \end{pmatrix}$$

(a) Example matrix.

Coordinate format for Sparse Matrix :

a_values = { 6, 9, 4, 4, 5, 3, 5, 8, 6, 5, 4, 3, 2, 2 }

i_index = { 0, 0, 0, 1, 2, 3, 3, 3, 4, 5, 6, 6, 7, 7 }

j_index = { 0, 2, 5, 5, 1, 2, 3, 4, 4, 5, 5, 6, 6, 7 }

Παραθέτουμε τον αλγόριθμο υπολογισμού του εσωτερικού γινομένου ένος αραιού πίνακα με διάνυσμα :

```
for j = 0 to j = M-1       // M = Num Rows of A array
        dot_product [j] = 0
        for  i = 0  to i = M-1
                dot_product[i_index[i]] += a_values[i] * b_vector[j_index[i]];
        end for i
end for j
```

Κόστος αλγορίθμου ώς προς N και nz(μη-μηδενικά στοιχεια) :

Κάθε στοιχείο του εσωτερικού γινομένου απαιτεί N πολλαπλασιασμούς-προσθέσεις και υπάρχουν nz στοιχεία. Έτσι το κόστος είναι :

$$O(N*nz)$$

Για τον σειριακό αλγόριθμο ισχύει πως για κάθε στοιχείο του dot_product απαιτούνται N πολ/σμοί και υπάρχουν nz στοιχεία. Έτσι ο σειριακός χρόνος εκτέλεσης είναι ίσος με

$$T1=tc * n2$$

(Όπου tc είναι ο χρόνος για ένα πολλαπλασιασμό-πρόσθεση.)

ΑΣΚΗΣΗ 2η

Για την υλοποίηση του παράλληλου αλγορίθμου για N στοιχεία σε p παράλληλες εργασίες πραγματοποιούμε διαχωρισμό του προβλήματος κατά block γραμμές (N/p) ανά επεξεργαστή. Χρησιμοποιούμε εντολές MPI,το οποίο είναι ένα σύστημα τυποποιημένο (Message Passing Interface), που καθιστούν δυνατή την επικοινωνία μεταξύ των επεξεργαστών και την αποστολή μεταξύ τους των επιμερούς αποτελεσμάτων με σκοπό την παράλληλη υλοποίση του αλγορίθμου και την βελτιστοποίηση της απόδοσης και του χρόνου εκτέλεσης.

Ακολουθούμε την ομαδοποίηση 1-D τεμαχισμού(Κατά γραμμές).Συνδιάζουμε n/p γραμμές από μικρές διεργασίες. Τα στοιχεία βρίσκονται αποθηκευμένα σε μία από τις διεργασίες και γίνεται broadcast σε όλες τις άλλες διεργασίες. Έχουμε **Συλλογική Επικοινωνία Εκπομπής(broadcast)**. Ο κόμβος πηγή στέλνει το ίδιο μήνυμα στους υπόλοιπους κόμβους (p-1). Κάθε διεργασία υπολογίζει το εσωτερικό γινόμενο των N/p γραμμών της με όλο το διάνυσμα x για να παράγει τις αντίστοιχες N/p σειρές του διανύσματος dot_product,οι οποίες αποθηκεύονται τοπικά.

Παραθέτουμε την υλοποίηση του παράλληλου αλγορίθμου:

```
Procedure mpi_sparsematrix_vector_mul

begin

N = M;

blocks = M / Numprocs;

Index = 0;
```

*row_start = MyRank * blocks;*

*row_end = (MyRank + 1) * blocks;*

(Υπολογίζονται τα επιμέρους εσωτερικά γινόμενα σε κάθε επεξεργαστή)

for i = 0 to i = nz-1

if i_index[i] < row_start

continue;

if i_index[i] >= row_end

continue;

*temp_product[i_index[i]] += a_values[i]*b_vector[j_index[i]];*

(Συλλογή στοιχείων στον επεξεργαστή Root)

*memcpy(dot_product, temp_product, M * sizeof(double));*

for i = 1 to i = Numprocs -1

MPI_Recv(temp_product, M,...)

for j = 0 to j = M-1

dot_product[j] += temp_product[j];

end for j

end for i

end procedure

<u>Κόστος Αλγόριθμου:</u>

Ο παραπάνω αλγόριθμος δημιουργεί ένα δένδρο για το δεδομένο δίκτυο με κορυφή την πηγή. Το κόστος διάδοσης μηνύματος Ν μεγέθους είναι :

$$O((p-1)nz+N)$$

Απόδοση :

- Ο χρόνος που απαιτείται για τους υπολογισμούς σε κάθε επεξεργαστή είναι:

$$Tcomp = tcNnz/p$$

- Ο χρόνος επικοινωνίας είναι :

$$Tcomm = (ts+tw*N/p)(p-1)$$

- Ο συνολικός χρόνος για τον παράλληλο αλγόριθμο είναι :

$$Tp = tc*N*nz/p + (ts + tw*N/p)(p-1) = tc*N*nz / p + ts*p + tw*N$$

- Η απόδοση είναι :

$$E(N,p) = T1/p*Tp = tc* N*nz / (tc*N*nz + ts*p^2 + tw*N*p)$$

Χρονοβελτίωση :

$$Sp = T1 / Tp = tc* N*nz / (tc*N*nz / p + ts*p + tw*N)$$

ΑΣΚΗΣΗ 4η: Sparse Matrix-Vector Multiplication(CSR Format)

A)Serial Algorithm

Το CSR (Compressed Sparse Row) format είναι ίσως το πιό ευρέως διαδεδομένο format για την συμπίεση ενός αραιού πίνακα,όταν δεν μας ενδιαφέρουν τα μή-μηδενικά στοιχεία που περιέχει. Αποτελείται από τρία διανύσματα {a_values,i_index,j_index}. Το double διάνυσμα a_values αποτελείται από τα μή-μηδενικά στοιχείου του αραιού πίνακα, το j_index περιέχει τον αριθμό της στήλης του αρχικού πίνακα όπου βρίσκεται το κάθε ένα στοιχείο nz(non zero) και το διάνυσμα i_index περίχει δείκτες στις θέσεις των διανυσμάτων a_values και j_index όπου ξεκινά η κάθε γραμμή του A. Έτσι τα στοιχεία της γραμμής i πίνακα A βρίσκονται στις θέσεις i_index[i] εώς i_index[i+1]-1.

Παράδειγμα αραιού Πίνακα και απεικόνηση του σε format CSR

$$\begin{pmatrix} 6 & 0 & 9 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 5 & 8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 2 \end{pmatrix}$$

(a) Example matrix.

array_vallues = {6 , 9 , 4 , 4 , 5 , 3 , 5 , 8 , 6 , 6 , 5 , 4 , 3 , 2 , 2 }

j_index = {1 , 3 , 6 , 6 , 2 , 3 , 4 , 5 , 5 , 6 , 6 , 7 , 7 , 8 }

i_ptr = {1 , 4 , 5 , 6 , 9 , 10 , 11 , 13 , 14 }

Παραθέτουμε τον αλγόριθμο υπολογισμού του εσωτερικού γινομένου ένος αραιού πίνακα με διάνυσμα

```
for i=0 to N-1
      dot_product[i] = 0;

      for j=i_index[i] to (i_index[i+1]-1)

          dot_product[i] = dot_product[i] + a_values[j-1] * b_vector[j_index[j-1]]

     end for j
end for i
```

<u>Κόστος αλγορίθμου:</u>

$$O(nz*N)$$