Sparse graph matrix reordering: Υλοποίηση αλγορίθμου Reverse Cuthill-McKee με openMP.

Φοιτητής: Μιχάλαινας Εμμανουήλ (9070)
michalain@ece.auth.gr

Καθηγητής: Πιτσιάνης Νικόλαος

Ημερομηνία: 13 Σεπτεμβρίου 2020

Περίληψη: Η παρούσα αναφορά παρουσιάζει τα σχόλια και τα συμπεράσματα που προέκυψαν από την εκπόνηση της 4ης εργασίας του μαθήματος “Παράλληλα και διανεμημένα συστήματα” - Τομέας Ηλεκτρονικής – Υπολογιστών – 7ο εξάμηνο. Ζητείται η επιλογή ενός εκ των αλγορίθμων Dulmage-Mendelsohn, Approximate minimum degree, Nested dissection και Reverse Cuthill McKee, η υλοποίηση μιας παράλληλης έκδοσης με Cilk ή OpenMP στη γλώσσα προγραμματισμού C και η σύγκριση του χρόνου εκτέλεσης με ήδη υπάρχουσες υλοποιήσεις. Σε αυτήν την εργασία επιλέχθηκε ο αλγόριθμος Reverse Cuthill McKee και υλοποιήθηκε η παράλληλη εκδοχή του με OpenMP, ενώ η σύγκριση γίνεται έναντι των συναρτήσεων symrcm που παρέχει το GNU Octave™ και το Matlab™. Ο κώδικας για την εργασία βρίσκεται στο github: https://github.com/manolismih/sparseMatPerm

Λέξεις-κλειδιά: C, OpenMP, parallel and distributed programming, high performance computing (HPC), thread, Cuthill McKee sparce graph matrix reordering.

Τμημα Ηλεκτρολογων Μηχανικων & Μηχανικων Υπολογιστων
Αριστοτελειο Πανεπιστημιο Θεσσαλονικης

1. ΣΥΝΤΟΜΑ ΣΧΟΛΙΑ ΚΑΙ ΠΑΡΑΤΗΡΗΣΕΙΣ

Παραγωγή πινάκων-εισόδων για τον αλγόριθμο

Το πρόγραμμα **generator.c** δέχεται ως μοναδικό όρισμα το μέγεθος n και δημιουργεί έναν τυχαίο συμμετρικό αραιό τετραγωνικό nxn πίνακα με στοιχεία 0 και 1, εκ των οποίων το ποσοστό των μη μηδενικών στοιχείων καθορίζεται από την παράμετρο FACTOR στον πηγαίο κώδικα. Ο πίνακας αποθηκεύεται στο αρχείο matrxix.txt.

Το σειριακό πρόγραμμα

Το πρόγραμμα **serial.c** αποτελεί την σειριακή έκδοση του αλγορίθμου reverse **Cuthill McKee**. Το πρόγραμμα διαβάζει τον πίνακα από το αρχείο **matrix.txt** και τυπώνει την μετάθεση που προκύπτει από την εκτέλεση του αλγόριθμου. Καθώς η είσοδος των δεδομένων είναι χρονοβόρα διαδικασία (και δεν περιλαμβάνεται στον χρόνο εκτέλεσης της συνάρτησης **symrcm**), ο **μετρητής χρόνου** ξεκινάει αφού αυτή τελειώσει.

Η υλοποίηση του αλγορίθμου γίνεται με τρόπο αποδοτικό ώστε να αποφεύγονται περιττές εγγραφές και αντιγραφές στην μνήμη. Καθώς σε κάθε βήμα του αλγορίθμου γίνονται ταξινομήσεις των ίδιων δεδομένων (ταξινόμηση κόμβων ανάλογα με τον βαθμό τους), είναι εφικτή η ταξινόμηση όλων των κόμβων στην αρχή, και έπειτα η δημιουργία ταξινομημένων **λιστών γειτνίασης** (χωρίς κλήση της qsort) για κάθε κόμβο. Το πλεονέκτημα είναι η αναδιάταξη των κόμβων με την σειρά που πρέπει να δεχτούν επίσκεψη από τον αλγόριθμο **BFS (Breadth-First Search)** και να εισαχθούν στην τελική μετάθεση.

Το παράλληλο πρόγραμμα

Με δεδομένη την έξυπνη σειριακή υλοποίηση, η παραλληλοποίηση εστιάζεται στα σημεία του σειριακού κώδικα που έχουν ένταση υπολογισμών. Τα σημεία αυτά είναι που έχουν **πολυπλοκότητα $O(n^2)$**, δηλαδή ο υπολογισμός του βαθμού κάθε κόμβου (άθροισμα κάθε γραμμής του πίνακα) και η δημιουργία των λιστών γειτνίασης. Τα σημεία αυτά είναι εύκολο να παραλληλισθούν, λόγω της απλότητας τους αλλά και λόγω του ότι δεν πραγματοποιούν εγγραφές σε ίδιες ή κοντινές θέσεις μνήμης.

Πράγματι, η απόπειρα παραλληλισμού των επαναλήψεων για την αρχικοποίηση του πίνακα perm, για την εύρεση της γραμμής με τον μικρότερο βαθμό και για την διάσχιση BFS οδηγεί σε **ασήμαντη βελτίωση ή ακόμα και σε καθυστέρηση του προγράμματος** σε σχέση με την σειριακή έκδοση! Αυτό γίνεται λόγω του φαινομένου **false sharing**: εγγραφές σε διαδοχικές θέσεις μνήμης που μοιράζονται την ίδια γραμμή στην κρυφή μνήμη οδηγούν σε συνεχείς ακυρώσεις των δεδομένων στους υπόλοιπους πυρήνες, για λόγους συνοχής της κρυφής μνήμης, ακόμα και όταν δεν υπάρχουν **data races.** Αυτός είναι και ο λόγος που στην παράλληλη έκδοση το άθροισμα κάθε γραμμής αθροίζεται σε τοπική μεταβλητή και όχι κατευθείαν στον πίνακα degree.

Matlab™ και GNU Octave™

Απλά εισάγουμε τα δεδομένα και μετρούμε τον χρόνο εκτέλεσης της symrcm:

```
A = importdata('matrix.txt',' ',1);
A = A.data;
p = symrcm(A);
```

2. ΑΠΟΤΕΛΕΣΜΑΤΑ

Οι δοκιμές έγιναν σε υπολογιστή με χαρακτηριστικά:

```
CPU: Intel Celeron N2830 2CPUs @ 2.16GHz
CACHE: L1d:48KB, L1i:64KB, L2:1MB
RAM: 4GB DDR3
```

n	GNU Octave™	Matlab™	Serial	Parallel
100	0.6 ms	0.5 ms	0.16 ms	0.35 ms
1000	64 ms	13 ms	5.34 ms	3.4 ms
10000	40 s	1.5 s	0.532 s	0.315 s

ΠΙΝΑΚΑΣ 1: Σύγκριση χρόνων εκτέλεσης των διαφορετικών υλοποιήσεων του αλγορίθμου Reverse CutHill-McKee για διαφορετικά μεγέθη του προβλήματος. Καθώς οι υλοποιήσεις των GNU Ocave™ και Matlab™ είναι κατά πολύ πιο αργές ακόμα και από την σειριακή υλοποίηση σε C, ουσιαστική σύγκριση μπορεί να γίνει μόνο μεταξύ της σειριακής και παράλληλης έκδοσης σε C.

Για μικρό μέγεθος προβλήματος n=100, η παράλληλη έκδοση είναι πιο αργή από την σειριακή, λόγω του επιπλέον κόστους σε χρόνο που απαιτείται για την έναρξη και διαχείριση των πολλαπλών νημάτων. Για μέτρια και μεγάλα, ωστόσο, μεγέθη προβλημάτων n=1000 και n=10000 είναι εμφανής η βελτίωση στην παράλληλη υλοποίηση. Οι επιταχύνσεις 1.57 και 1.69 αντίστοιχα είναι αρκετά ικανοποιητικές δεδομένης της ανώτερης θεωρητικά εφικτής επιτάχυνσης 2.00 που μπορεί να επιτευχθεί με το δεδομένο hardware.

3. ΑΝΑΦΟΡΕΣ

1. https://en.wikipedia.org/wiki/Cuthill%E2%80%93McKee_algorithm
2. https://www.mathworks.com/help/matlab/ref/symrcm.html
3. Ariful Azad, Mathias Jacquelin, Aydın Buluc¸, Esmond G. Ng, “The Reverse Cuthill-McKee Algorithm in Distributed-Memory” https://people.eecs.berkeley.edu/~aydin/RCM-ipdps17.pdf