Βασίλης Κυριαφίνης: 9797 Κόρο Έρικα: 9707

1ST ASSINGNMENT

parallel and destributed systems 2021-2022

Περιεχόμενα

[Εισαγωγή 1](#_Toc87923049)

[Σειριακός Αλγόριθμος 2](#_Toc87923050)

[Pthread library 3](#_Toc87923051)

[OpenMP API 4](#_Toc87923052)

# Εισαγωγή

Στόχος της παρούσης εργασίας ήταν η σύγκριση τεσσάρων μεθόδων εκτέλεσης ενός αλγορίθμου. Αρχικά, υλοποιήθηκε ο αλγόριθμος C σε σειριακή εκτέλεση για το ζητούμενο πρόβλημά μας και αξιοποιώντας αυτόν, με σκοπό την βελτίωση του χρόνου εκτέλεσής του, πραγματοποιήθηκαν τρεις ακόμη εκδοχές του με την χρήση των βιβλιοθηκών Pthread, OpenMP και OpenCilk.

Για την υλοποίηση των παραπάνω και για την αξιολόγησή τους εισήχθησαν πέντε γράφοι. Για τον κάθε γράφο υπολογίστηκε ο αριθμός των τριγώνων με την μέθοδο:

και

όπου Α ο αραιός πίνακας του κάθε γράφου, e ένας πίνακας στήλη με όλα τα στοιχεία του ίσα με τη μονάδα και c ο πίνακας του οποίου το κάθε στοιχείο υποδηλώνει σε πόσα τρίγωνα συμμετέχει ο αντίστοιχος κόμβος.

Να σημειωθεί ότι για την αποθήκευση των αραιών πινάκων στην μνήμη χρησιμοποιήθηκε η αναπαράσταση Compressed Sparse Row (CSR) η οποία όμως λόγω της συμμετρίας των πινάκων είναι ίδια με την Compressed Sparse Column (CSC).

Ο επεξεργαστής που χρησιμοποιήθηκε για τις δοκιμές είναι ο intel core i7-7700HQ [[1]](#footnote-1) ο οποίος διαθέτει 4 φυσικούς και 8 λογικούς πυρήνες. Το ρολόι είναι χρονισμένο στα 2.8 GHz και το Max Turbo Frequency είναι 3.8 GHz. Επίσης, διαθέτει 6 MB L3 cache, 4x256 Kbytes L2 cache, 4x32 Kbytes L1 instruction cache και 4x32 Kbytes L1 data cache.

Για το Compilation όλων των αρχείων πηγαίου κώδικα χρησιμοποιήθηκε η σημαία -O3 στον gcc με την οποία ο compiler παράγει εκτελέσιμα με βασικό γνώμονα τις επιδόσεις και όχι το τελικό μέγεθος του αρχείου.

# Σειριακός Αλγόριθμος

Ο σειριακός αλγόριθμος δέχεται ως είσοδο έναν πίνακα CSR, υπολογίζει τον πολλαπλασιασμό με τον εαυτό του εξαιρώντας τον υπολογισμό στις μηδενικές θέσεις του αραιού πίνακα. Με τον τρόπο αυτό υπολογίζεται αυτόματα και το “προϊόν” Hadamard. Στη συνέχεια αθροίζονται οι τιμές όλων των στοιχείων του παραγόμενου CSR πίνακα και διαιρώντας με το έξι προκύπτει ο αριθμός των τριγώνων του αρχικού γράφου.

Ο πολλαπλασιασμός του CSR πίνακα με τον εαυτό του γίνεται με ένα διπλό for loop στο οποίο μία μία οι γραμμές του πίνακα πολλαπλασιάζονται με τις στήλες του. Το αποτέλεσμα αποθηκεύεται κατευθείαν σε μορφή CSR.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Belgium\_osm | com\_YouTube | Mycielskian13 | LAW | NACA0015 |
| Time | 79.267 ms | 4.887 s | 1.236 s | 153.826 ms | 699.452 ms |

Τα παραπάνω αποτελέσματα όπως και τα αποτελέσματα που θα παρουσιαστούν παρακάτω αποτελούν τον μέσο όρο 10 εκτελέσεων του αλγορίθμου για κάθε έναν από τους γράφους. Επιλέχθηκε αυτός ο τρόπος παρουσίασης των τελικών αποτελεσμάτων γιατί οι χρόνοι εκτέλεσης διαφέρουν από εκτέλεση σε εκτέλεση, λόγω εξωτερικών του προγράμματος παραγόντων όπως άλλα προγράμματα, προγραμματισμένες διεργασίες του λειτουργικού συστήματος κ.ά.

# Pthread library

Για την πρώτη παράλληλη υλοποίηση χρησιμοποιήθηκε η βιβλιοθήκη Pthread. Στην πορεία, δοκιμάστηκαν διάφορες τεχνικές παραλληλοποίησης μεταξύ των οποίων και τα thread pools, τα οποία προσφέρουν καλύτερη κατανομή του συνολικού φόρτου ανάμεσα στα threads αλλά λόγω της έντονης ανάγκης για συγχρονισμό τους, η απόδοση δεν ήταν η αναμενόμενη. Τελικά, ως καλύτερη λύση βρέθηκε η πλήρης ανεξαρτητοποίηση των threads μεταξύ τους ώστε να μην υπάρχει ανάγκη συγχρονισμού όσο αυτά εκτελούνται.

Ο αλγόριθμος που χρησιμοποιήθηκε είναι κατά βάση ο ίδιος με την σειριακή υλοποίηση. Η κατανομή της εργασίας στα thread έγινε σε επίπεδο γραμμών του πίνακα (εξωτερική for), δηλαδή το κάθε thread ήταν υπεύθυνο να υπολογίσει το γινόμενο n γραμμών με όλες τις στήλες του πίνακα όπου,

# OpenMP API

1. <https://ark.intel.com/content/www/us/en/ark/products/97185/intel-core-i77700hq-processor-6m-cache-up-to-3-80-ghz.html> [↑](#footnote-ref-1)