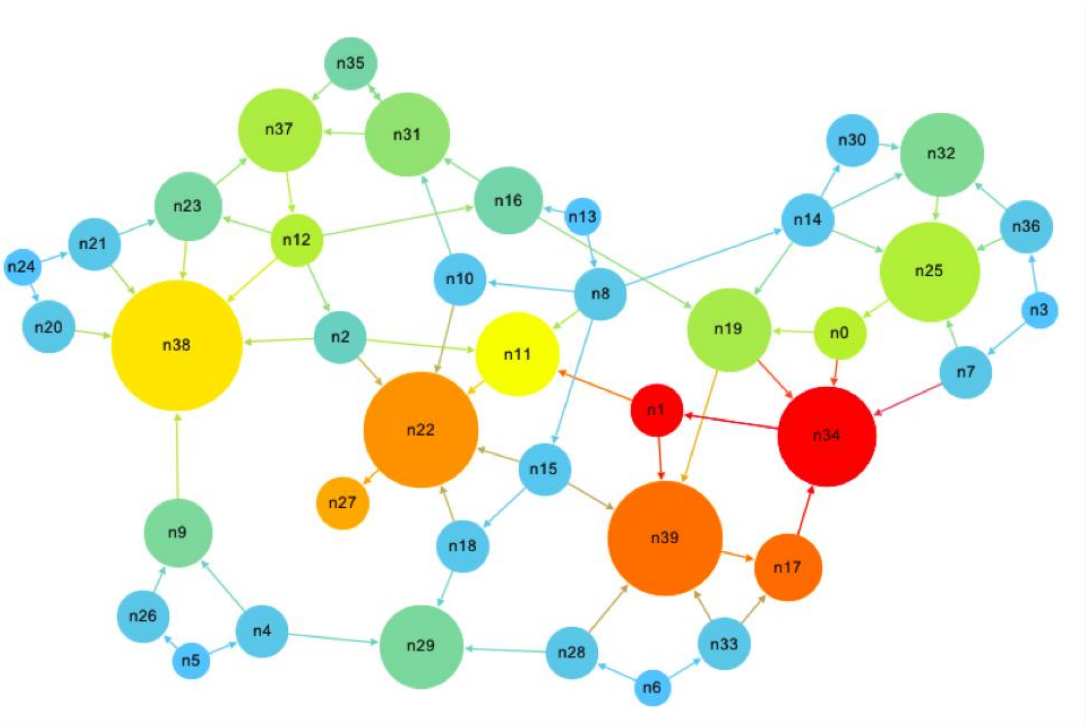
Εργασία #4

Parallel PageRank Algorithm



ΜΟΥΡΟΥΖΗ ΧΡΙΣΤΟΣ 7571

ΑΓΡΟΤΗΣ ΙΩΑΝΝΗΣ 7567

Σκοπός της εργασίας:

Σκοπός της εργασίας είναι η παράλληλη υλοποιήση σε POSIX threads του αλγόριθμου PageRank της Google ο οποίος μετρά τον αριθμό και την ποιότητα των διασυνδέσεων (edges) σε έναν κόμβο (node) με σκόπο να καθορίσει την σημαντικότητα του κόμβου αυτού.

Η PageRank είναι ένας μαθηματικός αλγορίθμος βασισμένος στα web graphs που έχουν δημιουργηθεί από ιστοσελίδες του World Wide Web, οι οποίες λειτουργούν ως κόμβοι και υπερσυνδέσμους ως ακμές. Η rank τιμή σηματοδοτεί την σπουδαιότητα μιας συγκεκριμένης ιστοσελίδας-κόμβου. Το PageRank μίας ιστοσελίδας-κόμβου καθορίζεται επαναληπτικά και εξαρτάται από το πλήθος των διασυνδέσεων της, καθώς και το PageRank αυτών των διασυνδέσεων. Έτσι ένας κόμβος που διασυνδέεται με πολλούς άλλους με υψηλό PageRank λαμβάνει και ο ίδιος ένα υψηλό PageRank. Ο τύπος ο οποίος χρησιμοποιείται για τον υπολογισμό του PageRank είναι:

$$P_i^{t+1} = d \sum_{j \in B_i} P_j^t \frac{\alpha_{ij}}{\sum_{k \in B_j} \alpha_{jk}} + (1-d)E_i$$

Όπου:

α) P_i^{t+1} η πιθανότητα να βρεθεί στον κόμβο i την χρονική στιγμή $t+1$.

β) P_i^t η πιθανότατα ο χρήστης να βρίσκεται στον κόμβο j τη χρονική στιγμή t.

γ) α_{ij} μεταβλητή που παίρνει τιμές $[1, 0]$ ανάλογα με την ύπαρξη ή μη σύνδεσης μεταξύ των κόμβων i, j.

δ) B_i, B_j τα σύνολα των κόμβων με τους οποίους οι i, j συνδέονται.

ε) e_i η πιθανότητα ο χρήστης να βρεθεί στον κόμβο i μέσω κάποιας τυχαίας αναζήτησης.

ζ) $\sum_{k \in B_j} \alpha_{jk}$ ο βαθμός συνδεσιμότητας του κόμβου j.

στ) d παράγοντας που καθορίζει τη σημασία του κάθε όρου της εξίσωσης, συνήθης τιμή $d = 0.85$.

Περιγραφή Διαδικασίας:

Η παραλληλοποίηση του κώδικα έγινε με χρήση των POSIX threads, ο σειριακός του οποίου βασίστηκε στον κώδικα MATLAB που μας δόθηκε.

Τα dataset που χρησιμοποιήθηκαν περιείχαν ως δεδομένα ζευγάρια διασυνδέσεων μεταξύ των διάφορων κόμβων. Οι κόμβοι της κάθε διασύνδεσης είναι διαχωρισμένοι από τον χαρακτήρα 9 του κώδικα ASCII (TAB), ενώ οι διασυνδέσεις διαχωρίζονται με τον χαρακτήρα ‘’\n’’. Η διαδικασία εισαγωγής δεδομένων στον αλγόριθμο γίνεται μέσω της συνάρτησης **import_data()**.

Στην συνέχεια δημιουργούνται threads , τα οποία τρέχουν την συνάρτηση **kernel** παράλληλα. Κάθε thread αναλαμβάνει την επεξεργασία συγκεκριμένου αριθμού κόμβων, αριθμός οποίος ισούται με τον αριθμό των συνολικών κόμβων δια τον αριθμό των threads.

Η συνάρτηση **kernel** εκτελεί τα εξής βήματα:

Αρχικά, για κάθε κόμβο που αναλογεί στο εκάστοτε thread, γίνεται αναζήτηση μέσα στον πίνακα διασυνδέσεων(E) προκειμένου να βρεθεί ένας πίνακας(L) που περιέχει τους κόμβους που συνδέονται με το συγκεκριμένο κόμβο.

Στη συνέχεια, αφού έχουν υπολογιστεί οι πίνακες διασυνδέσεων όλων των κόμβων, το πρόγραμμα προχωρά σε υπολογισμό του PageRank κάθε κόμβου. Οι υπολογισμοί αυτοί επιτυγχάνονται μέσω διαδοχικών επαναλήψεων μέχρι να σταθεροποιηθούν.

Έλεγχος Ορθότητας:

Για τον έλεγχο ορθότητας του κώδικα αρχικά εκτελέσαμε την συνάρτηση pagerank_demo.m του MATLAB, συνάρτηση η οποία μέσω της surfer.m δημιούργησε ένα dataset 100 κόμβων από το site www.amazon.com και στην συνέχεια υπολόγισε το

PageRank καθενός από αυτούς τους κόμβους. Το παραπάνω dataset τροφοδοτήθηκε (αφού το μετατρέψαμε σε κατάλληλη μορφή) στον παράλληλο κώδικα που γράψαμε. Μετά από σύγκριση των αποτελεσμάτων του σειριακού κώδικα MATLAB και του παράλληλου διαπιστώθηκε ότι τα αποτελέσματα προκύπτουν ίδια.

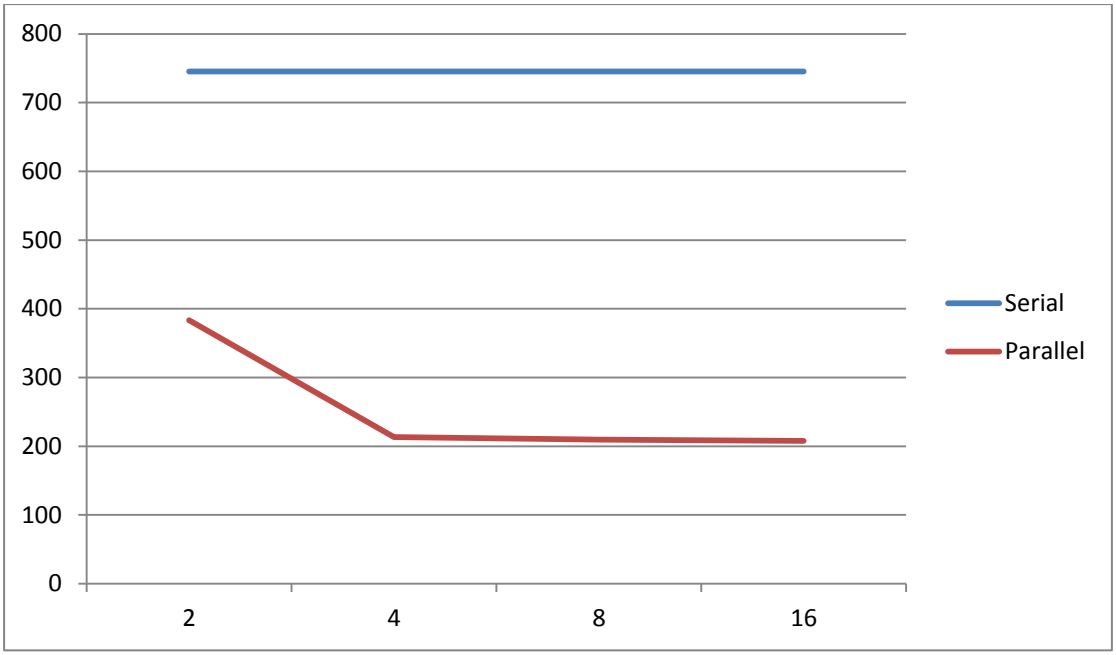
Δεδομένα:

Τα dataset που χρησιμοποιήθηκαν για την εκτέλεση του αλγορίθμου βρέθηκαν στην ιστοσελίδα http://snap.stanford.edu/data/. Συγκεκριμένα (αφού αφαιρέσαμε τα σχόλια που είναι γραμμένα στην αρχή των αρχείων txt) χρησιμοποιήσαμε τα εξής:

- Amazon0302.txt (nodes = 262111, edges = 1234877)
- Amazon0601.txt (nodes = 403394, edges = 3387388)
- web-Google.txt (nodes = 875713, edges = 5105039)

Σύγκριση χρόνων εκτέλεσης σειριακού και παράλληλου κώδικα:

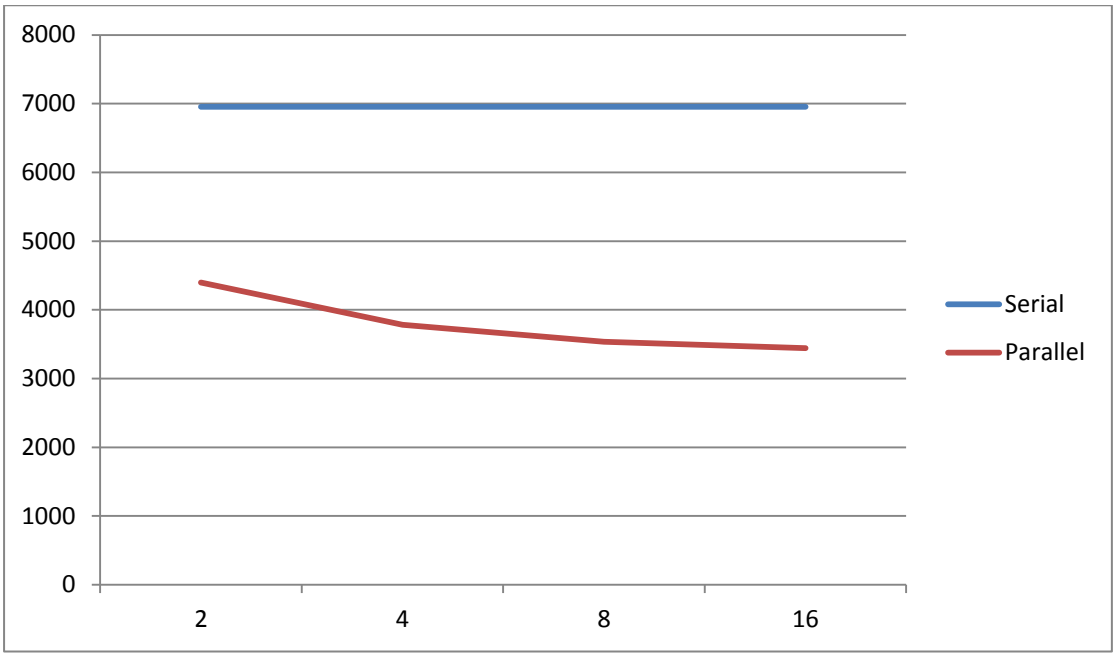
1. **Amazon0302** nodes = 262111, edges = 1234877



Σειριακός (ms)	745,37	745,37	745,37	745,37
Νήματα	**2**	**4**	**8**	**16**
Παράλληλος (ms)	**383,54**	**213,24**	**209,73**	**207,82**
Ταχύτητα	**x1,94**	**x3,49**	**x3,55**	**x3,58**

Παρατηρήσεις: Από τα αποτελέσματα βλέπουμε ότι ο παράλληλος κώδικας είναι πιο γρήγορος από το σειριακό. Μεγάλη διαφορά παρουσιάζεται από τα 2 στα 4 νήματα όπου σχεδόν διπλασιάζεται η ταχύτητα και έπειτα σταθεροποιείται για 8 και 16 νήματα αντίστοιχα.

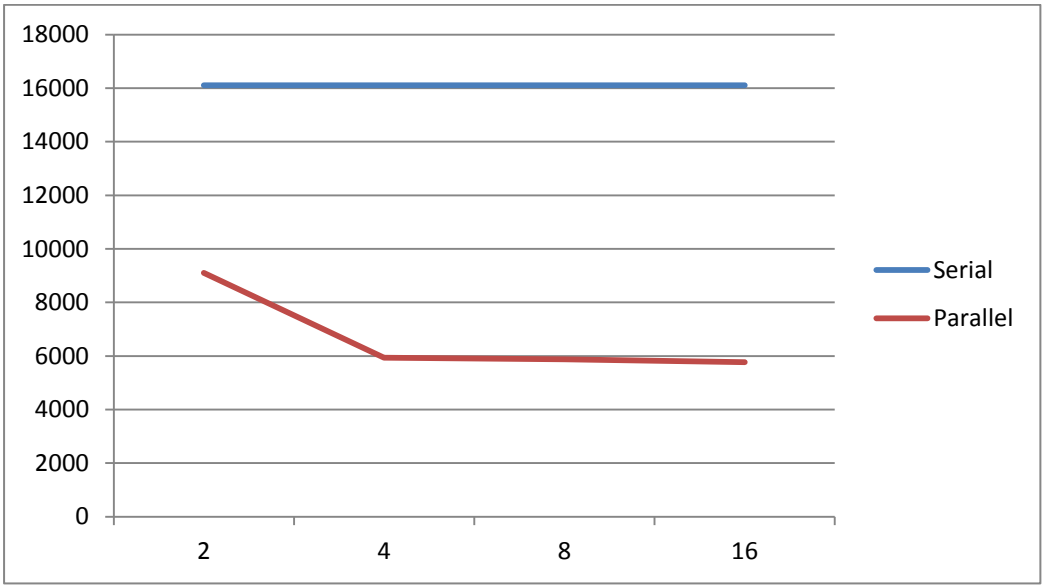
2. **Amazon0601** nodes = 403394, edges = 3387388



Σειριακός (ms)	6952,5	6952,5	6952,5	6952,5
Νήματα	**2**	**4**	**8**	**16**
Παράλληλος (ms)	**4396,5**	**3780,4**	**3533,9**	**3441,3**
Ταχύτητα	**x1,58**	**x1,83**	**x1,96**	**x2,02**

Παρατηρήσεις: Από τα αποτελέσματα βλέπουμε ότι ο παράλληλος κώδικας είναι πιο γρήγορος από το σειριακό. Μικρή διαφορά παρουσιάζεται από τα 2 στα 4 νήματα όπου σχεδόν διπλασιάζεται η ταχύτητα και έπειτα σταθεροποιείται για 8 και 16 νήματα αντίστοιχα.

3. **Web-google** nodes = 875713, edges = 5105039



Σειριακός (ms)	16103,7	16103,7	16103,7	16103,7
Νήματα	**2**	**4**	**8**	**16**
Παράλληλος (ms)	**9098,2**	**5934,1**	**5872,2**	**5767,8**
Ταχύτητα	**x1,76**	**x2,71**	**x2,74**	**x2,79**

Παρατηρήσεις: Από τα αποτελέσματα βλέπουμε ότι ο παράλληλος κώδικας είναι πιο γρήγορος από το σειριακό. Μεγάλη διαφορά παρουσιάζεται από τα 2 στα 4 νήματα όπου σχεδόν διπλασιάζεται η ταχύτητα και έπειτα σταθεροποιείται για 8 και 16 νήματα αντίστοιχα.

Γενικές Παρατηρήσεις:

Η ταχύτητα εκτέλεσης του παράλληλου κώδικα δεν διπλασιάζεται ή τετραπλασιάζεται για 2 και 4 thread αντίστοιχα, σε σχέση με το σειριακό, όπως θα ήταν αναμενόμενο. Αυτό οφείλεται, εν μέρει, στο ότι το 2ο κομμάτι της συνάρτησης kernel, που υπολογίζει τις πιθανότητες (PageRank), πραγματοποιεί πρόσβαση σε κοινές (μεταξύ των thread) μεταβλητές, γεγονός που καθιστά κρίσιμο το τμήμα αυτό του κώδικα. Αυτό έχει ως συνέπεια το συγκεκριμένο αυτό κομμάτι να εκτελείται σχεδόν σειριακά.