Σταματάκης Ε Στυλιανός

ΑΜ:4041

ΑΣΚΗΣΗ 1

Να αρχίσω λέγοντας ότι εχω δυσλεξία. Οπότε αν γραφώ κάτι που δεν βγάζει πολύ νόημα, και γενικά άμα έχετε κάποιο πρόβλημα με την βαθμολόγηση μπορείτε να επικοινωνήσετε μαζί μου ώστε να σας εξηγήσω καλυτέρα τι έκανα. Ευχαριστώ!

Σχετικά με το πως γίνεται το RUN και το Compile, μπορείτε να ρίξετε μια ματιά στο Makefile που έκανα. Το έκανα όσο μπορούσα self-explanatory. Αλλά γενικά η ιδέα είναι ότι για να πάνε όλα καλά ο χρήστης πρέπει να δώσει πρώτα το όνομα του άρχει και μετα των αριθμό τον threads που θέλει. Οποιαδήποτε άλλη προσπάθεια σε είσοδο θα φάει assert.

πχ ./exec.out readme.txt 2 → σωστό!

Σχετικά με την υλοποίηση εχω τρία αρχεία. Τα δυο είναι ένας header και η υλοποίηση του και μετα εχω την main. Μέσα στην main έχει μονάχα την main και το function που θα τρέχουν τα/το thread και τίποτα άλλο. Στο header file εχω την υπογραφεί των συναρτήσεων που χρησιμοποιώ και τα structs. Περιληπτικά η ιδέα της υλοποίησης μου είναι ότι κάθε φορά που διαβάζω ένα νέο node, για να δούμε άμα είναι νέο node πρέπει να πάμε να κοιτάξουμε την λίστα με ήδη υπάρχων nodes O(n), το δημιουργό και φτιάχνω μια λίστα με τους γείτονες του. Τα Node της λίστας αυτής της λίστας έχουν ένα node pointer οπού δείχνει στο σωστό node που είναι ο γείτονας μου. Αφού διαβάσω ολοι το αρχείο φτιάχνω έλα lookup table με pointer σε Nodes. Εκεί αποθηκεύω τους pointer και διαγραφώ/free την δυναμική λίστα που είχα όταν τους έκανα εισαγωγή. Έπειτα φτιάχνω τον σωστό αριθμό threads και κάνω κλήση της συνάρτησης η οποία κάνει τα έξεις. Σε κάθε iteration κάθε thread κάνει δυο φάσεις. Πρώτα πάει και βρίσκει για κάθε κόμβο που το αντιστοιχεί ποσό πρέπει να δώσει από το PageRank του στους γείτονες του. Και αφού έχουν τελειώσει όλα τα thread με αυτό(barrier) πάνε στην δεύτερη φάση οπού πάνε στους γείτονες κάθε κόμβου και τους δίνουν το PageRank που είχαμε υπολογίσει πριν, ΟΜΩΣ επειδή πολύ μπορεί να δείχνουν σε ένα κόμβο και μπορεί να έχουμε ταυτόχρονη πρόσβαση στον ίδιο κόμβο πρέπει να κάνουμε lock τον κόμβο που θέλουμε να αυξήσουμε το PageRank του. Σε αντίθεση με την πρώτη φάση οπού δεν έχουμε αυτόν τον φόβο καθώς ξέρουμε ότι στον πίνακα είναι κάθε στοιχείο μια φορά και δεν γίνεται να το πειράξουν δυο threads ταυτόχρονα. Αφού λοιπόν τελειώσει και η δεύτερη φάση, τα threads περιμένουν να τελειώσουν ολοι(barrier) και πάνε στο επόμενο iteration. Τέλος τα γράφουμε στο αρχείο και κάνουμε free ό,τι space έχουμε κάνει malloc.

Για τις παρακάτω μετρήσεις δεν το έτρεξα ακριβώς 5 φορές το έτρεξα 7 - 8 καθώς μερικές φορές τα αποτελέσματα δεν βοηθούσαν να φανεί το speedup.

- File: p2p-Gnutella24.txt

For <u>one</u> thread:		**For <u>three</u> thread:**	
First run:	0m5.365s	First run:	0m5.305s
Second run:	0m5.367s	Second run:	0m5.300s
Third run:	0m5.369s	Third run:	0m5.295s
Forth run:	0m5.362s	Forth run:	0m5.296s
Fifth run:	0m5.363s	Fifth run:	0m5.296s
Average:	*0m5.365s*	*Average:*	*0m5.298s*
Deviation:	*0.002561*	*Deviation:*	*0.003720*
Speed Up:	*0*	*Speed Up:*	*0m0.018s*
For <u>tow</u> thread:		**For <u>four</u> thread:**	
First run:	0m5.315s	First run:	0m5.286s
Second run:	0m5.312s	Second run:	0m5.288s
Third run:	0m5.321s	Third run:	0m5.282s
Forth run:	0m5.318s	Forth run:	0m5.293s
Fifth run:	0m5.318s	Fifth run:	0m5.281s
Average:	*0m5.316s*	*Average:*	*0m5.286s*
Deviation:	*0.003059*	*Deviation:*	*0.004335*
Speed Up:	*0m0.049s*	*Speed Up:*	*0m0.012s*

- File: Email-Enron.txt

For <u>one</u> thread:		**For <u>three</u> thread:**	
First run:	1m13.745s	First run:	1m12.716s
Second run:	1m13.157s	Second run:	1m12.861s
Third run:	1m13.728s	Third run:	1m11.208s
Forth run:	1m11.005s	Forth run:	1m12.687s
Fifth run:	1m12.843s	Fifth run:	1m12.723s
Average:	*1m12.895s*	*Average:*	*1m12.439s*
Deviation:	*1.005965*	*Deviation:*	*0.618439*
Speed Up:	*0*	*Speed Up:*	*0m0.118s*
For <u>tow</u> thread:		**For <u>four</u> thread:**	
First run:	1m12.861s	First run:	1m12.616s
Second run:	1m11.734s	Second run:	1m12.692s
Third run:	1m12.861s	Third run:	1m11.015s
Forth run:	1m12.976s	Forth run:	1m12.192s
Fifth run:	1m12.357s	Fifth run:	1m11.410s
Average:	*1m12.557s*	*Average:*	*1m11.985s*
Deviation:	*0.464266*	*Deviation:*	*0.665189*
Speed Up:	*0m0.338s*	*Speed Up:*	*0m0.466s*

- File: facebook_combined.txt

For <u>one</u> thread:		**For <u>three</u> thread:**	
First run:	0m0.887s	First run:	0m0.811s
Second run:	0m0.885s	Second run:	0m0.812s
Third run:	0m0.883s	Third run:	0m0.808s
Forth run:	0m0.876s	Forth run:	0m0.809s
Fifth run:	0m0.885s	Fifth run:	0m0.809s
Average:	*0m0.883s*	*Average:*	*0m0.809s*
Deviation:	*0.003815*	*Deviation:*	*0.001469*
Speed Up:	*0*	*Speed Up:*	*0m0.018s*
For <u>tow</u> thread:		**For <u>four</u> thread:**	
First run:	0m0.825s	First run:	0m0.803s
Second run:	0m0.827s	Second run:	0m0.804s
Third run:	0m0.825s	Third run:	0m0.803s
Forth run:	0m0.833s	Forth run:	0m0.803s
Fifth run:	0m0.827s	Fifth run:	0m0.806s
Average:	*0m0.827s*	*Average:*	*0m0.803s*
Deviation:	*0.002939*	*Deviation:*	*0.001166*
Speed Up:	*0m0.056s*	*Speed Up:*	*0m0.006s*