Σταματάκης Ε Στυλιανός

ΑΜ:4041

ΑΣΚΗΣΗ 2

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 τις μέρες που θέλει, δεύτερο το όνομα του αρχείου που έχει τους αρρώστους κόμβους,μετα το όνομα του αρχείου που φτιαχνει το γραφο και τέλος των αριθμό τον threads που θέλει.

πχ ./exec.out 30 readme1.txt 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 νέα nodes, για να δούμε άμα είναι νέα nodes πρέπει να πάμε να κοιτάξουμε την λίστα με τα ήδη υπάρχων nodes, δημιουργω αυτα που πρεπει και φτιάχνω την 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 τσεκαρει ποιοι κομβοι κολησαν, αλλάζοντας ένα FLAG, απο κάθε κόμβο που είναι αρρωστος. Έπειτα γίνεται έλεγχος εάν πεθαίνει η αν είναι η ώρα να αναρρώσει. Και αφού έχουν τελειώσει όλα τα thread με αυτό(barrier) πάνε στην δεύτερη φάση οπού όντως είναι το τέλος της μέρας και οι κομβόι που πρέπει να γίνουν άρρωστοι αλλάζουν το FLAG τους. Και στις δυο φάσεις δεν έχουμε φόβο για rase conditions 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.

Γενικά πειραματίστηκα πολύ με τον τρόπο που θα γίνεται παράλληλο το πρόγραμμα όμως τελικά κατέληξα στην πιο απλή λύση καθώς είχε και την καλύτερη απόδοση από όλους τους άλλους. Γιατρό και εγώ άφησα τα tasks και στα for αν και προσπάθησα διαφορά schedules αλλά δεν βελτίωσε την απόδοση.

Επίσης να πω ότι μερικά από τα ζητήσουνε της αναφοράς με μπέρδεψαν αρκετά αλλά νομίζω ότι εν τέλη τα κατάφερα. Εάν κάτι δεν βγάζει νόημα μπορείτε να επικοινωνήσετε μαζί μου ώστε να σας εξηγήσω. Μέτρησα μόνο τον χρόνο εκτέλεσης της παράλληλης συνάρτησης και όχι όλης της εκτέλεσης όπως την προηγουμένη άσκηση για να φαίνεται καλυτέρα η μείωση χρόνου.

Το overhead του OpenMP προγράμματος με παραλληλισμό 1, σε σχέση με το σειριακό πρόγραμμα (δηλαδή μεταγλωττισμένο χωρίς -fopenmp) οφείλετε στο κόστος να φτιάχτουν τα threads και να γίνει ο συγχρονισμός τους. Ενώ όταν έχουμε ένα δεν υπάρχουν αυτοί οι extra υπολογισμοί. Ακόμα ανάλογα τον κώδικα και το scheduling υπάρχει extra κόστος. Πχ εκκίνηση του thread, ανάθεση υπολογισμών κτλ.

Το speedup δεν το επιρεαζει τόσο ο αριθμος των επαναλήψεων (30-365), καθως το παραλληλο κομματι δεν έχει να κανει με το ποσος μέρες ειναι. Αυτό όμως που αξιζει να σημειωθεί ειναι οτι σε καθε iteration φτιάχνουμε κάθε φορά νέα threads οπότε με αυτή την εννιά επηρεάζει κάπως το speedup. Βέβαια το openmp χρησιμοποιεί ένα μηχανισμό που λέγετε thread pool για να αποφιγει περιπτοσεις σαν αυτές. Στο thread pool αποθηκευονται όλα τα created threads που είναι σε inactive state. Οπότε όταν ζητούμουν θα δοθούν εκείνα και δεν θα φτιάξει νέα το σύστημα. Τώρα σχετικά με το μέγεθος του αρχείου υπάρχουν πολλές περιπτώσεις. Όμως σιγουρά παίζει σημασία στο speedup του προγράμματος καθώς έχει να κάνει με overheads όπως loading balance, εκκίνηση και σταμάτημα των threads, scheduling. . Αν έχουμε πχ ένα γραφτός που φτιάχτηκε από το input είναι πολύ μικρος όμως εμείς έχουμε πολλά threads λογο των παραπάνω κοστών δεν θα δούμε το προγραμμα να τρέχει πιό γρηγορα. Οπότε ανάλογα το input file πρέπει να γίνουν οι σωστές επιλογές για να έχουμε καλή απόδοση.

- File: p2p-Gnutella24.txt

For <u>zero</u> thread:		**For <u>tow</u> thread:**	
Day30:		Day30:	
Average Speed Up:	*0,003676*	*Average Speed Up:*	*0,003058*
Average Throughput:	*8161,04*	*Average Throughput:*	*9810,33*
Διακύμανση:	*1,476E-10*	*Διακύμανση:*	*5,42521E-07*
Day180:		Day180:	
Average Speed Up:	*0,022642*	*Average Speed Up:*	*0,020447*
Average Throughput:	*7949,83*	*Average Throughput:*	*8803,25*
Διακύμανση:	*8,0752E-09*	*Διακύμανση:*	*5,1848E-07*
Day365:		Day365:	
Average Speed Up:	*0,046199*	*Average Speed Up:*	*0,042903*
Average Throughput:	*7900,60*	*Average Throughput:*	*8507,56*
Διακύμανση:	*4,10174E-08*	*Διακύμανση:*	*2,61009E-07*
For <u>one</u> thread:		**For <u>four</u> thread:**	
Day30:		Day30:	
Average Speed Up:	*0,004085*	*Average Speed Up:*	*0,002064*
Average Throughput:	*7343,94*	*Average Throughput:*	*14534,88*
Διακύμανση:	*1,5376E-10*	*Διακύμανση:*	*1,20978E-06*
Day180:		Day180:	
Average Speed Up:	*0,024807*	*Average Speed Up:*	*0,019209*
Average Throughput:	*7256,02*	*Average Throughput:*	*9370,61*
Διακύμανση:	*1,21244E-08*	*Διακύμανση:*	*8,66824E-06*
Day365:		Day365:	
Average Speed Up:	*0,047510*	*Average Speed Up:*	*0,040649*
Average Throughput:	*7682,59*	*Average Throughput:*	*8979,31*
Διακύμανση:	*7,9762E-08*	*Διακύμανση:*	*4,65584E-06*

- File: Email-Enron.txt

For <u>zero</u> thread:		**For <u>tow</u> thread:**	
Day30:		Day30:	
Average Speed Up:	*0,005149*	*Average Speed Up:*	*0,004333*
Average Throughput:	*5826,37*	*Average Throughput:*	*6923,61*
Διακύμανση:	*3,004E-10*	*Διακύμανση:*	*9,91781E-07*
Day180:		Day180:	
Average Speed Up:	*0,032782*	*Average Speed Up:*	*0,030255*
Average Throughput:	*5490,82*	*Average Throughput:*	*5949,43*
Διακύμανση:	*6,40918E-08*	*Διακύμανση:*	*9,57454E-07*
Day365:		Day365:	
Average Speed Up:	*0,066581*	*Average Speed Up:*	*0,065256*
Average Throughput:	*5482,04*	*Average Throughput:*	*5593,36*
Διακύμανση:	*1,02995E-07*	*Διακύμανση:*	*1,83344E-07*
For <u>one</u> thread:		**For <u>four</u> thread:**	
Day30:		Day30:	
Average Speed Up:	*0,005645*	*Average Speed Up:*	*0,003487*
Average Throughput:	*5314,44*	*Average Throughput:*	*8603,38*
Διακύμανση:	*2,03656E-09*	*Διακύμανση:*	*3,46506E-06*
Day180:		Day180:	
Average Speed Up:	*0,033381*	*Average Speed Up:*	*0,030891*
Average Throughput:	*5392,29*	*Average Throughput:*	*5826,94*
Διακύμανση:	*4,33578E-08*	*Διακύμανση:*	*1,45338E-05*
Day365:		Day365:	
Average Speed Up:	*0,068950*	*Average Speed Up:*	*0,063219*
Average Throughput:	*5293,69*	*Average Throughput:*	*5773,58*
Διακύμανση:	*3,17434E-08*	*Διακύμανση:*	*3,73092E-06*

- File: facebook_combined.txt

.

For <u>zero</u> thread:		**For <u>tow</u> thread:**	
Day30:		Day30:	
Average Speed Up:	*0,000540*	*Average Speed Up:*	*0,000480*
Average Throughput:	*55555,56*	*Average Throughput:*	*62500,00*
Διακύμανση:	*5,896E-11*	*Διακύμανση:*	*1,784E-11*
Day180:		Day180:	
Average Speed Up:	*0,032786*	*Average Speed Up:*	*0,031587*
Average Throughput:	*5490,15*	*Average Throughput:*	*5698,55*
Διακύμανση:	*3,7584E-10*	*Διακύμανση:*	*1,8279E-06*
Day365:		Day365:	
Average Speed Up:	*0,064587*	*Average Speed Up:*	*0,062115*
Average Throughput:	*5651,29*	*Average Throughput:*	*5876,20*
Διακύμανση:	*1,09018E-08*	*Διακύμανση:*	*1,17646E-06*
For <u>one</u> thread:		**For <u>four</u> thread:**	
Day30:		Day30:	
Average Speed Up:	*0,000701*	*Average Speed Up:*	*0,000389*
Average Throughput:	*42796,01*	*Average Throughput:*	*77120,82*
Διακύμανση:	*2,304E-11*	*Διακύμανση:*	*9,69726E-08*
Day180:		Day180:	
Average Speed Up:	*0,035934*	*Average Speed Up:*	*0,032274*
Average Throughput:	*5009,18*	*Average Throughput:*	*5577,24*
Διακύμανση:	*6,9736E-10*	*Διακύμανση:*	*1,74788E-06*
Day365:		Day365:	
Average Speed Up:	*0,068538*	*Average Speed Up:*	*0,061166*
Average Throughput:	*5325,51*	*Average Throughput:*	*5967,37*
Διακύμανση:	*1,6948E-09*	*Διακύμανση:*	*1,94794E-06*