HY-342 Άσκηση 1 (Parallel Page Rank)

Φοίβος Παπαπαναγιωτάκης, csd3823.

Αρχεία κώδικα

- ParalPageRank.c – περιέχει τον παράλληλο αλγόριθμο, την δομή του γράφου και την main().
- symtable.h & symtablehash.c – είναι μία βιβλιοθήκη που είχα φτιάξει στα πλαίσια του HY - 255. Είναι ένα lookup table.

Περιγραφή προγράμματος

Δέχεται ένα αρχείο ως input, το οποίο αποτελείται από τις ακμές ενός κατευθυνόμενου γράφου (ζεύγη κόμβων).

Για κάθε κόμβο γίνεται η εξής διαδικασία: αναζητείται σε ένα lookup table. Εάν βρέθηκε το lookup table θα μας επιστρέψει το index που έχει αυτός ο κόμβος στην δομή του γράφου, εάν όχι, θα προστεθεί στη δομή του γράφου και στο lookup table.

Αφού και οι δύο κόμβοι βρίσκονται σίγουρα στην δομή του γράφου, προστίθεται σε μία λίστα του “from” κόμβου, το id (στη δομή του γράφου – όχι το real id) του “to” κόμβου.

Όταν τελειώσει η ανάγνωση του αρχείου γνωρίζουμε ότι ο πίνακας (δομή του γράφου) δεν θα γίνει ξανά realloc() και άρα αντικαθιστούμε, στην λίστα ακμών κάθε κόμβου, το id με pointer. Επίσης διαγράφεται και το lookup table.

Στη συνέχεια, μοιράζονται οι κόμβοι του γράφου ίσα στα threads και ξεκινάνε.

How to compile / run

Η μεταγλώττιση γίνεται με ένα “make all”.

Για την εκτέλεση υπάρχουν 2 σημαντικά πράγματα:

1. Το argv[1] είναι ο αριθμός των threads για την εκτέλεση του προγράμματος. Πρέπει να είναι >= 1 και <= 4. Σε περίπτωση που μείνει κενό ένα thread θα δημιουργηθεί.
2. Το input γίνεται με **redirection**

Οπότε μία εκτέλεση για το αρχείο “graph.txt” με 4 threads θα ήταν

./a.out 4 < graph.txt

Αποτελέσματα (ταχύτητα)

Ο υπολογισμός χρόνου εκτέλεσης έγινε στα μηχανήματα της σχολής με το time (time ./a.out 4 < input.txt). Το output έγινε με redirection.

p2p-Gnutella24:

- 1 thread: 0.202s, 0.190s, 0.188s, 0.197s, 0.202s
  - Avg = 0.1958s
  - Standard deviation = 0.0058s
- 2 threads: 0.158s, 0.167s, 0.157s, 0.156s, 0.156s
  - Avg = 0.1588s
  - Standard deviation = 0.0041s
- 3 threads: 0.156s, 0.134s, 0.137s, 0.137s, 0.135s
  - Avg = 0.1398s
  - Standard deviation = 0.0082s
- 4 threads: 0.127s, 0.120s, 0.153s, 0.169s, 0.137s
  - Avg = 0.1412s
  - Standard deviation = 0.0178s

Average speedups:

- 1 -> 2 threads: 0.037s
- 2 -> 3 threads: 0.019s
- 3 -> 4 threads: -0.0014

ego-Facebook:

- 1 thread: 0.173s, 0.152s, 0.184s, 0.165s, 0.167s
  - Avg = 0.1682s
  - Standard deviation = 0.0105s
- 2 threads: 0.122s, 0.118s, 0.118s, 0.118s, 0.117s
  - Avg = 0.1186s
  - Standard deviation = 0.0017s
- 3 threads: 0.101s, 0.103s, 0.102s, 0.104s, 0.100s
  - Avg = 0.102s
  - Standard deviation = 0.0014s
- 4 threads: 0.112s, 0.105s, 0.104s, 0.100s, 0.093s
  - Avg = 0.1028s
  - Standard deviation = 0.0062s

Average speedups:

- 1 -> 2 threads: 0.0496s
- 2 -> 3 threads: 0.0166s
- 3 -> 4 threads: -0.0008s

email-Enron:

- 1 thread: 0.795s, 0.789s, 0.795s, 0.774s, 0.760s
  - Avg = 0.7826s
  - Standard deviation = 0.0137s
- 2 threads: 0.693s, 0.706s, 0.745s, 0.687s, 0.698s
  - Avg = 0.7058s
  - Standard deviation = 0.0206s
- 3 threads: 0.641s, 0.678s, 0.690s, 0.659s, 0.648s
  - Avg = 0.6632s
  - Standard deviation = 0.0183s
- 4 threads: 0.657s, 0.628s, 0.622s, 0.711s, 0.636s
  - Avg = 0.6508s
  - Standard deviation = 0.0323s

Average speedups:

- 1 -> 2 threads: 0.0768s
- 2 -> 3 threads: 0.0426s
- 3 -> 4 threads: 0.0124s

Παραλληλισμός και συγχρονισμός

Για n threads οι κόμβοι του γράφου χωρίζονται σε 1 / n τμήματα.

Το κάθε thread θα διατρέξει τους κόμβους που του αντιστοιχούν. Στον καθένα από τους κόμβους θα διατρέξει την λίστα από κόμβους στους οποίους δείχνει και θα μοιράσει το (0.85 * 1 / κ) του rank του, όπου κ το πλήθος κόμβων στους οποίους δείχνει. Όταν μοιράζει το rank ο κόμβος που το δέχεται κλειδώνει ένα mutex.

Αφού τελειώσουν όλα τα threads αυτή την διαδικασία (barrier 1) ο αλγόριθμος περνάει στην επόμενη φάση του, όπου θέτεται το καινούργιο rank όλων των κόμβων.

Στην συνέχεια αφού τελειώσουν όλα τα threads (barrier 2) την ανανέωση του rank ξεκινάει πάλι η πρώτη φάση.