Παράλληλα και Διανεμημένα Συστήματα
Εργασία - 4 Pagerank

Κυριαζής Γεώργιος - Λέανδρος gkyriazt@ece.auth.gr 7711

September 29, 2018

1 Συμπεριλαμβανόμενα αρχεία

Link: Dropbox

φάκελος **Final** με όλα τα αρχεία κώδικα ήτοι:

- folder Implementation_Ver1
- folder Implementation_Ver2
- Report_Project_4_Parallel.pdf
- Report_Project_4_Parallel.tex

Ο φάκελος Implementaion 1 περιέχει υλοποίηση του Pagerank με αποθήκευση δεδομένων διατηρώντας τον Α τετράγωνο και κρατώντας όλα τα μηδενικά σπαταλώντας μνήμη. Αντίθετα στην υλοποίηση 2 τα δεδομένα αποθηκεύονται συμπιεσμένα εξοικονομώντας μνήμη έτσι δίνεται η δυνατότητα διαχείρισης πιο πολλών δεδομένων. Στο κάθε φάκελο υπάρχουν τα αποτελέσματα μετρήσεων καθώς και οι αντίστοιχοι κώδικες. Τέλος θα γίνει μια προσπάθεια σύγκρισης της μεθόδου Gauss-Siedel με την Power Method η οποία είναι εξίσου δημοφιλής.

2 Implementation_1

Σαν οδηγό στην υλοποίηση αλλά και σαν έλεγχο ορθότητας χρησιμοποιώ αυτό το Paper . Αυτό το paper λοιπόν παρουσιάζει ένα δίκτυο 4 κόμβων και σαν τελικό Pagerank δίνει το παρακάτω διάνυσμα.

$$P_p = \begin{bmatrix} 0.1194 & 0.3314 & 0.2602 & 0.2890 \end{bmatrix} \tag{1}$$

ενώ ο δικός μου αλγόριθμος δίνει σαν έξοδο το παρακάτω διάνυσμα

$$P_{mine} = \begin{bmatrix} 0.12005499 & 0.33244571 & 0.25729886 & 0.29020050 \end{bmatrix} \tag{2}$$

εφόσον οι τιμές είναι πολύ κοντά θεωρώ ότι ο αλγόριθμος δίνει σωστά αποτελέσματα ειδικά αν συνυπολογίσει κανείς το πλήθος των κόμβων αυτές οι διαφορές είναι μάλλον ανούσιες.

2.1 Παρουσίαση Αλγορίθμου

Ο αλγόριθμος παίρνει σαν όρισμα το όνομα του αρχείου τον αριθμό των κόμβων το d που γενικά είναι ίσο με 0.85 καθώς και tolerance δηλαδή την ελάχιστη επιτρεπτή διαφορά στο διάνυσμα pagerank μεταξύ δύο διαδοχικών επαναλήψεων του αλγορίμθου. Αρχικά διαβάζεται το αρχείο και δημιουργείται ο πίνακας A οπου a[i][j] σημαίνει ότι υπάρχει link από τον κόμβο i στον κόμβο j. Έπειτα με την βοήθεια συναρτήσεων αρχικοποιείται το P όπου κάθε στοιχείο παίρνει την τιμή 1/n οπού n ο αριθμός των κόμβων. Μετά μετρώνται πόσα link ξεκινάν από τον κάθε κόμβο με την συνάρτηση count_outlinks. Ύστερα ό Α στοχαστικοποίειται μέσω της συνάρτησης stochastifyMatrix και τέλος η συνάρτηση createMMatrix χειρίζεται την περίπτωση των dangling nodes. Έπειτα ξεκινάει ο αλγόριθμος Pagerank αφου αρχικοποιηθεί το διάνυσμα P_new όπου θα αποθηκεύονται οι νέες τιμές που προκύπτουν από το γινόμενο A*P. Φυσικά σύμφωνα με το Gauss Siedel όταν υπολογίζεται ένα στοιχείο P_new[i] χρησιμοποιείται στον υπολογισμό των υπολοίπων στοιχείων και αυτό όντως κάνει την υλοποίση να συγκλίνει γρηγορότερα. Προφανώς μόλις πιάσουμε το tolerance ο αλγόριθμος σταματάει. Επειδή η υλοποίηση δεν απορρίπτει τα μηδενικά (και εφόσον έχω δώσει 40GB στο σύστημά μου) τρέχει δίκτυα μέχρι το πολύ 10k κόμβους. Ως εκ τούτου χρησιμοποιήθηκαν 2 δίκτυα το Wiki-Vote με 8794 κόμβους και το P2P-Gnutella με 8114.

2.2 Παρουσίαση αποτελεσμάτων

Παρακάτω παρουσιάζονται διάφορα αποτελέσματα εκτέλεσης του αλγορίθμου. στο κουτάκια με την slash (/) ο ακέραιος αριθμός είναι ο αριθμός των επαναλήψεων ώς την σύγκλιση και ο δεκαδικός είναι ο χρόνος εκτέλεσης σε δευτερόλεπτα (s).

Wiki-Vote / Error	Gs_serial	GS_Parallel	PowerMeth_Par
10^{-4}	17 / 7.6000	14 / 3.8516	26 / 5.8100
10^{-8}	34 / 14.6800	32 / 8.6800	55 / 12.1700
P2P-Nutella / Error			
10^{-4}	9 / 3.4300	8 / 1.8566	14 / 2.6500
10^{-8}	16 / 6.1300	16 / 3.7500	32 / 8.6800

Παρατηρούμε πως όσο το tolerance μειώνεται τόσο αυξάνει ο αριθμός των επαναλήψεων ως την σύγκλιση αλλά και ο συνολικός χρόνος. Επίσης παρατηρούμε πως η μέθοδος gauss Siedel όντως συγκλίνει και πιο γρήγορα αλλά και σε λιγότερο αριθμό επαναλήψεων. Γενικά στην περίπτωση αυτή όντως η παραλληλοποίση βελτίωσε τον χρόνο εκτέλεσης αισθητά.

3 Implementation_2

Σε σύγκριση με το παραπάνω paper το διάνυσμα που προκύπτει από αυτόν τον αλγόριθμο είναι το παρακάτω:

$$P_{mine} = \begin{bmatrix} 0.1691332012414932 & 0.3488371968269348 & 0.2410147935152054 & 0.2410147935152054 \end{bmatrix} \tag{3}$$

και πάλι είναι αρκετά κοντά με το αρχικό οπότε είμαστε ικανοποιημένοι.

3.1 Παρουσίαση Αλγορίθμου

Σε αυτή την υλοποίηση αποθηκεύσαμε το δίκτυο ώς compact sparse matrix με αποτέλεσμα να μπορούμε να τρέχουμε δίκτυα μέχρι και 800k κόμβους. Αρχικά ο πίνακας val έχει παντού τίμη 1 και έπειτα παίρνει την τιμή 1/outlinks[i] ανάλογα με το πόσα link φεύγουν από τον κάθε κόμβο. Το col ind διάνυσμα μας δείχνει προς τα που δείχνει ο κάθε κόμβος και το row_ptr μας δείνχει που τελειώνει ο ένας κόμβος και αρχίζουν τα outlinks του επόμενου. Έπειτα εκτελούμε τον αλγόριθμο ανάλογα με το και πάλι αν κάποιο p_new[i] στοχείο έχει υπολογιστει το χρησιμοποιούμε στον υπολογισμό των υπολοίπων σύμφωνα με την μέθοδο Gauss-Siedel.

3.2 Παρουσίαση αποτελεσμάτων

dataset / Error	Nodre-dam	Google	BerkStan	Stanford
10^{-6}	6 / 0.09	3 / 0.04	4 / 0.187	3 / 0.038
10^{-11}	52 / 0.71	3 / 0.04	5 / 1.95	4 / 0.477

Παρατηρούμε εδώ ότι οι χρόνοι είναι αρκετά μικροί. Πολύ πιθανό να βοηθάει το γεγόνος ότι αποθηκεύοντας τον πίνακα ως compact sparse να περιορίζει εκθετικά τον αριθμό των πράξεων με αποτέλεσμα να πέφτει δραματικά ο χρόνος. Παρόλα αυτά οποιαδήποτε προσπάθεια παραλληλοποίησης(πέρα από τα προφανή σημεία) του αλγορίθμου απέτυχε. Πιθανόν αυτό να συμβαίνει εξαιτίας του overhead μεταξύ των threads.

4 Επιρροές

Η εργασία μου επιρρεάστηκε από τις δουλειές των παρακάτω ανθρώπων:
Link: Link_1

Link: Link_2
Link: Link_3