ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

«Αποδοτικός Υπολογισμός Ερωτημάτων σε Δυναμικά Μεταβαλλόμενους Γράφους με την Χρήση του Timely Dataflow»

Εξαμηνιαία Εργασία

για το μάθημα «Ανάλυση και Σχεδιασμός Πληροφοριακών Συστημάτων»

Χαράλαμπος Δάλπης el17067
Αντώνιος Καραντώνης el17439
Δημήτριος Κυριακίδης el17077

Διδάσκων: Δ. Τσουμάκος

ΠΕΡΙΛΗΨΗ

Στην παρούσα εργασία εξετάζουμε το σύστημα differential dataflow, μια βιβλιοθήκη βασισμένη στο μοντέλο Naiad timely dataflow, ως προς την απόδοσή του σε αλγορίθμους γράφων, εφαρμοζόμενους σε δυναμικώς μεταβαλλόμενα δεδομένα. Συγκεκριμένα, πραγματοποιούμε σειρά πειραμάτων, με την υλοποίηση του συστήματος αυτού στην γλώσσα Rust, δοκιμάζοντας διαφορετικές τοπολογίες και διαστάσεις γράφων, διαφορετικούς όγκους μεταβολών αυτών, και διαφορετικούς συνήθεις αλγορίθμους. Στο τέλος εξάγουμε συμπεράσματα επί των αποτελεσμάτων των πειραμάτων αυτών.

«Λέξεις - κλειδιά»: Naiad, Timely Dataflow, Differential Dataflow, αλγόριθμοι γράφων σε δυναμικά δεδομένα, triangle counting, Pagerank, degree centrality, μέτρηση χρόνου εκτέλεσης

Όλα τα αρχεία κώδικα και αποτελέσματα της εργασίας έχουν δημοσιευθεί στο εξής github repository: https://github.com/DimK19/big-data-project. Αναλυτικά όλες οι μετρήσεις, και τα μεγέθη και χρόνοι κατασκευής των γράφων που χρησιμοποιήθηκαν, στο αρχείο excel "big_data_results.xlsx".

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Τα τελευταία χρόνια το Διαδίκτυο κατακλύζεται καθημερινά από πολύ μεγάλο όγκο δεδομένων. Η αποδοτική διαχείριση των δεδομένων αυτών αποτελεί μεγάλη πρόκληση για την επιστήμη της πληροφορικής, αλλά και επιτακτική ανάγκη. Τα σύγχρονα υπολογιστικά συστήματα αξιοποιούν διάφορα υπολογιστικά μοντέλα για την επεξεργασία «μεγάλων δεδομένων». Δύο ενδιαφέροντα μοντέλα, τα οποία υποστηρίζουν επαναληπτικούς και συνεχείς υπολογισμούς σε δυναμικά δεδομένα είναι το Timely Dataflow και το Differential Dataflow.

Σκοπός

Σκοπός της παρούσας εργασίας είναι η εφαρμογή των υπολογιστικών μοντέλων Timely Dataflow και Differential Dataflow σε αλγορίθμους γράφων και η εκτίμηση της απόδοσής τους, όσον αφορά την ταχύτητα των υπολογισμών. Επίσης, θα εξεταστεί το πως μεταβάλλεται ο χρόνος εκτέλεσης των αλγορίθμων κατά την μεταβολή του συνόλου δεδομένων του γράφου.

Συγκεκριμένα, χρησιμοποιούνται οι open-source εκδόσεις σε Rust των παραπάνω μοντέλων, ενώ εκτελούνται ενδεικτικά πάνω στους γράφους οι παρακάτω υπολογιστικοί αλγόριθμοι: ο αλγόριθμος εύρεσης τριγώνων σε μη κατευθυνόμενους γράφους (triangle counting), ο αλγόριθμος pagerank σε κατευθυνόμενους γράφους και ο αλγόριθμος υπολογισμού των βαθμών κορυφών σε μη κατευθυνόμενους γράφους (degree centrality).

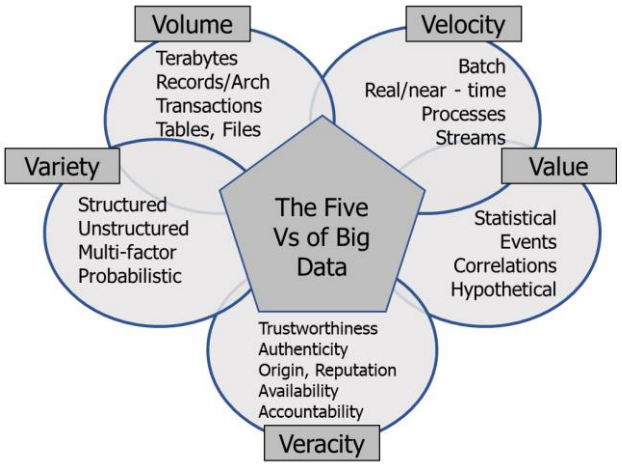
ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

Μεγάλα Δεδομένα

Ο όρος Μεγάλα Δεδομένα (Big Data) αναφέρεται σε σύνολα δεδομένων που είναι τόσο μεγάλα ή σύνθετα που ξεφεύγουν από τις δυνατότητες καταγραφής, αποθήκευσης και ανάλυσης με παραδοσιακές μεθόδους. Τα μεγάλα δεδομένα αναφέρονται σε μη δομημένα, ημι-δομημένα και δομημένα δεδομένα, κυρίως όμως εστιάζουν στα μη δομημένα δεδομένα[1].

Τα μεγάλα δεδομένα προέρχονται από την κυκλοφορία δεδομένων στον ιστό, τα μηνύματα ηλεκτρονικού ταχυδρομείου, το περιεχόμενο των μέσων κοινωνικής δικτύωσης, αλλά και τα αυτόματα παραγόμενα στοιχεία δεδομένων από αισθητήρες. Οι οργανισμοί συλλέγουν και αναλύουν μεγάλα δεδομένα με στόχο να βελτιώσουν τις διαδικασίες τους, αλλά και τις αποφάσεις που λαμβάνουν.

Τα μεγάλα δεδομένα περιγράφονται από τα παρακάτω χαρακτηριστικά, γνωστά και ως 5Vs:



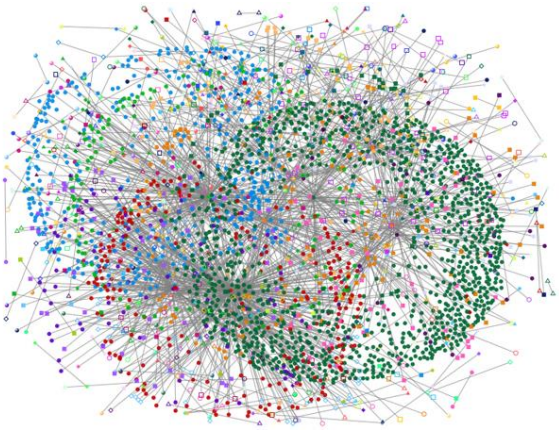
- Όγκος (Volume)
  Αναφέρεται στο μέγεθος του συνόλου των δεδομένων και καθορίζει το αν θα χαρακτηριστούν «μεγάλα».
- Ταχύτητα (Velocity)
  Αναφέρεται στην ταχύτητα με την οποία παράγονται, αποθηκεύονται και επεξεργάζονται. Πολλές φορές μάλιστα απαιτείται η ανάλυσή τους σε πραγματικό χρόνο (real time).
- Ποικιλία (Variety)
  Αναφέρεται στους πολλούς διαφορετικούς τύπους των δεδομένων (κείμενα, εικόνες, ηχητικά μηνύματα, video, κλπ. Οι σύνδεση και συσχέτιση των δεδομένων αυτών μεταξύ τους μπορεί να οδηγήσει σε χρήσιμα συμπεράσματα.
- Αλήθεια (Veracity)
  Τα μεγάλα δεδομένα έχουν νόημα μόνο αν είναι ακριβή και αξιόπιστα.
- Αξία (Value)

Τα μεγάλα δεδομένα έχουν αξία μόνο αν μπορούμε να μεταφέρουμε την αξία αυτή σε εφαρμογές στον πραγματικό κόσμο.

Μία πολύ συνηθισμένη δομή δεδομένων που χρησιμοποιείται στην ανάλυση Μεγάλων Δεδομένων είναι οι γράφοι, είτε με κατευθυνόμενες, είτε με μη-κατευθυνόμενες ακμές, καθώς μπορούν να χρησιμοποιηθούν για την μοντελοποίηση και περιγραφή διαφόρων προβλημάτων.

Γράφοι

Ο γράφος ή γράφημα (graph) είναι μια αφηρημένη αναπαράσταση ενός συνόλου στοιχείων, και κατ' επέκτασιν μία γενική μορφή δομής δεδομένων όπου μερικά ζεύγη στοιχείων συνδέονται μεταξύ τους με δεσμούς. Τα διασυνδεδεμένα στοιχεία ονομάζονται κόμβοι ενώ οι δεσμοί που συνδέουν τα ξεύγη των κορυφών ονομάζονται ακμές.



Εικ. 1: παράδειγμα μη-κατευθυνόμενου γράφου

Ένας γράφος *G*, με σύνολο κόμβων *V* και σύνολο ακμών *E* συμβολίζεται *G(V,E)*. Οι ακμές μπορούν να είναι κατευθυνόμενες (ασύμμετρες) ή μη-κατευθυνόμενες (συμμετρικές). Συνήθως, ένα γράφημα απεικονίζεται σε διαγραμματική μορφή ως ένα σύνολο κουκκίδων για τις κορυφές, ενωμένα μεταξύ τους με γραμμές για τις ακμές. [3], [4]

Αλγόριθμοι γράφων

Ένας αλγόριθμος γράφου είναι ένα σύνολο εντολών με βάση τις ακμές και τις κορυφές του γράφου, που οδηγεί στην επίλυση προβλημάτων που αφορούν την θεωρία γράφων.

Λόγω της σημαντικότητας και της ευρείας χρήσης των γράφων ως δομές δεδομένων, οι αλγόριθμοι αυτοί έχουν άμεσες εφαρμογές στην ανάλυση Μεγάλών Δεδομένων.

Εμείς θα ασχοληθούμε συγκεκριμένα με τρεις αλγορίθμους, τον αλγόριθμο εύρεσης τριγώνων (triangle counting), τον αλγόριθμο pagerank και τον αλγόριθμο υπολογισμού βαθμών κορυφών (degree centrality).

Αλγόριθμος εύρεσης τριγώνων (triangle counting)

Ο αλγόριθμος εύρεσης τριγώνων υπολογίζει το πλήθος των τριγώνων που υπάρχουν σε έναν μη-κατευθυνόμενο γράφο. Ως τρίγωνο ορίζουμε ένα σύνολο τριών κόμβων, όπου κάθε ένας από αυτούς έχει ακμή με κάθε έναν από τους άλλους δύο. Έχει χρονική πολυπλοκότητα της τάξης του $O(|V|^3)$, όπου $|V|$ το πλήθος των κόμβων του γράφου. [13]

Αλγόριθμός pagerank

Ο αλγόριθμος pagerank υπολογίζει την σημαντικότητα του κάθε κόμβου ενός κατευθυνόμενου γράφου, με βάση το πλήθος των ακμών που προσπίπτουν στον κόμβο αυτόν (έσω-ακμές). Έχει χρονική πολυπλοκότητα της τάξης του $O(|V|+|E|)$ όπου $|V|$ το πλήθος των κόμβων και $|E|$ το πλήθος των ακμών του γράφου. [11]. Αναλυτική περιγραφή της λειτουργίας του στο παράρτημα.

Αλγόριθμος υπολογισμού των βαθμών κορυφών (degree centrality)

Ο αλγόριθμος υπολογισμού βαθμών κορυφών υπολογίζει για κάθε κόμβο ενός μη-κατευθυνόμενου γράφου το πλήθος των ακμών που προσπίπτουν σε αυτόν. Χρησιμοποιείται για την εύρεση των πιο «διάσημων» κόμβων, δηλαδή των κόμβων που εμφανίζουν τις περισσότερες σχέσεις (ακμές) με άλλους κόμβους του γράφου. Έχει χρονική πολυπλοκότητα της τάξης του $O(V^2)$. [6]

Μοντέλο Timely Dataflow

Το Timely Dataflow είναι ένα υπολογιστικό μοντέλο ροής δεδομένων (dataflow framework), το οποίο παρουσιάστηκε για πρώτη φορά στο άρθρο “Naiad: a Timely Dataflow System” [9]. Το μοντέλο αυτό είναι ικανό να διαχειρίζεται και να εκτελεί υπολογισμούς επί μεγάλου όγκου δεδομένων σε παράλληλες ροές, μειώνοντας τον χρόνο εκτέλεσης.
Βασίζεται σε έναν κατευθυνόμενο γράφο στον οποίο οι κορυφές, που αναπαριστούν μία κατάσταση, στέλνουν και λαμβάνουν μηνύματα με λογική χρονική σήμανση κατά μήκος κατευθυνόμενων ακμών. Μια σημαντική ιδιότητα του μοντέλου Timely Dataflow είναι ότι χρησιμοποιεί νήματα (worker threads) για την εκτέλεση των απαιτούμενων διεργασιών. Οι workers δημιουργούνται κατά την εκτέλεση του προγράμματος και ο

καθένας αναλαμβάνει την επεξεργασία διαφορετικού μέρους των δεδομένων. Εμείς χρησιμοποιούμε μια εκτεταμένη και πιο αρθρωτή (modular) εφαρμογή του Timely Dataflow η οποία είναι υλοποιημένη σε Rust. [8]

Μοντέλο Differential Dataflow

Το Differential Dataflow είναι επίσης ένα υπολογιστικό μοντέλο, το οποίο είναι δομημένο με βάση το Timely Dataflow, το οποίο χρησιμοποιείται για την εκτέλεση υπολογισμών σε μεγάλο όγκο ταχέως μεταβαλλόμενων δεδομένων. [7] Μπορούμε, δηλαδή, αφού γράψουμε και εκτελέσουμε το πρόγραμμά μας, να εφαρμόσουμε αλλαγές στην είσοδό του και το Differential Dataflow θα υπολογίσει και θα εμφανίσει τις αλλαγές στην έξοδο σε συγκριτικά πάρα πολύ μικρό χρόνο. Τις ανωτέρω λειτουργίες και πλεονεκτήματα αυτού του υπολογιστικού μοντέλου θα αποπειραθούμε να αξιοποιήσουμε, εκτελώντας τα κατάλληλα πειράματα.

ΜΕΘΟΔΟΛΟΓΙΑ

Υπολογιστικό Σύστημα

Το υπολογιστικό σύστημα που θα χρησιμοποιήσουμε για την εκτέλεση των πειραμάτων διαθέτει εξαπύρηνο επεξεργαστή AMD Ryzen 5 5500U με 12 threads, ο οποίος είναι χρονισμένος στα 2.1GHz, και 16GB μνήμης RAM DDR5. Έχει εγκατεστημένη την έκδοση Arch Linux βασισμένη στο λειτουργικό σύστημα Linux, το οποίο προτιμήσαμε από μία αντίστοιχη έκδοση των Windows, καθώς επιτρέπει καλύτερη διαχείριση των διαθέσιμων πόρων του συστήματος.

Επιπλέον, στα προγράμματά μας χρειάζεται να δημιουργηθούν και μετέπειτα να φορτωθούν γράφοι οι οποίοι είναι πολύ μεγάλοι σε μέγεθος, με αποτέλεσμα η υπάρχουσα φυσική μνήμη RAM να μην είναι επαρκής. Για να ανταποκριθούμε επομένως στις υψηλές ανάγκες που έχει η εκτέλεση του κώδικα σε μνήμη χρησιμοποιήσαμε την εναλλαγή μνήμης (memory swapping), δεσμεύοντας επιπλέον 130GB χώρου στον δίσκο για την εκτέλεση των διεργασιών μας, η οποία είναι ωστόσο σημαντικά χαμηλότερη σε ταχύτητα. Αυτό γίνεται εμφανές στους καταγεγραμμένους χρόνους εκτέλεσης, όπου παρουσιάζεται αλματώδης διαφορά μεταξύ εκτελέσεων όπου χρειάστηκε και δεν χρειάστηκε η εναλλαγή. Την αρνητική επίδραση της αναγκαίας αυτής τεχνικής στους χρόνους υπολογισμού την λαμβάνουμε υπ’ όψιν στον σχολιασμό των αποτελεσμάτων.

Δημιουργία προγράμματος αυτοματοποιημένης παραγωγής γράφων

Για την εκτέλεση των πειραμάτων χρειάστηκε ως είσοδο να χρησιμοποιηθούν γράφοι οι οποίοι φέρουν συγκεκριμένα χαρακτηριστικά, όπως για παράδειγμα το μέγεθός τους ή ο τρόπος σύνδεσης των κόμβων μεταξύ τους με ακμή. Για τον λόγο αυτόν, χρειάστηκε να κατασκευάσουμε ένα εργαλείο με το οποίο να μπορούμε συστηματικά να δημιουργούμε γράφους με τα επιθυμητά χαρακτηριστικά και να ορίσουμε τις μεταβολές που πρόκειται να πραγματοποιηθούν στις ακμές τους.

Πρόγραμμα σε Python

Αρχικά δημιουργήσαμε με την χρήση της γλώσσας Python το εργαλείο `graph_gen.py`, το οποίο χρησιμοποιεί την βιβλιοθήκη της Python *NetworkX*. Το εργαλείο αυτό δέχεται ως είσοδο το είδος του γράφου που θέλουμε να δημιουργήσουμε (random, S-F) και τα επιμέρους χαρακτηριστικά για τον συγκεκριμένο γράφο (όπως πλήθος κόμβων, πιθανότητα σύνδεσης δύο κόμβων) και η έξοδος που παράγει είναι δύο αρχεία. Το αρχείο `random_graph.txt` σε κάθε γραμμή του έχει ένα ζεύγος κόμβων χωρισμένο με κενό διάστημα, το οποίο ορίζει μια ακμή του γράφου που μόλις δημιουργήθηκε. Το αρχείο `edges_to_change.txt` σε κάθε γραμμή του έχει ένα ζεύγος κόμβων χωρισμένο με κενό διάστημα και δίπλα έναν εκ των αριθμών 1 και -1. Το ζεύγος κόμβων ορίζει την ακμή του γράφου που πρόκειται να επεξεργαστούμε και ο αριθμός το αν πρόκειται για προσθήκη ή αφαίρεση ακμής (1 για προσθήκη και -1 για αφαίρεση). Κατά την παραγωγή του αρχείου των μεταβολών έχει γίνει πρόβλεψη ώστε να προστίθενται μόνο ακμές που δεν υπάρχουν και να αφαιρούνται μόνο ακμές που υπάρχουν.
Το εργαλείο αυτό διαθέτει ένα σύνολο επιπλέον λειτουργιών που αφορούν τις αλλαγές που μπορούμε να εφαρμόσουμε σε έναν γράφο που δημιουργήσαμε. Υπάρχει δυνατότητα να αφαιρέσουμε έναν ολόκληρο κόμβο από τον γράφο (αφαιρώντας μία προς μία όλες τις προσπίπτουσες ακμές). Επίσης μπορούμε να αφαιρέσουμε ή και να προσθέσουμε ακμές οι οποίες έχουν ιδιαίτερη σημασία για τον γράφο (π.χ. αφαιρούμε ακμές που έχουν υψηλή πιθανότητα να συμμετέχουν σε πολλά τρίγωνα, αφαιρούμε πολλές ακμές από έναν κόμβο μειώνοντας σημαντικά τον βαθμό του)

Βελτίωση και πρόγραμμα σε C++

Η βιβλιοθήκη *NetworkX* όμως της Python για να δημιουργήσει τους ζητούμενους γράφους, κρατάει αποθηκευμένη στην μνήμη κάθε ακμή που προσθέτει μέχρι να ολοκληρωθεί η διαδικασία, με αποτέλεσμα σε μεγάλους γράφους να γεμίζει την μνήμη και να μην καταφέρνει να τους δημιουργήσει επιτυχώς. Για τον λόγο αυτόν, προχωρήσαμε στην κατασκευή δύο νέων εργαλείων με την χρήση της γλώσσας C++, του `graph.cpp` και του `graph_changes.cpp`.

Το πρώτο δημιουργεί το αρχείο `random_graph.txt` με τον γράφο που χρειαζόμαστε με τα ζητούμενα χαρακτηριστικά, χωρίς όμως να χρειάζεται να αποθηκεύει στην μνήμη τις ακμές που προσθέτει κάθε φορά.

- Συγκεκριμένα, για τους τυχαίους γράφους το πετύχαμε εύκολα δημιουργώντας απλά τυχαίες ακμές με δύο εμφωλευμένες επαναλήψεις (χωρίς να απαιτείται η διατήρηση του συνολικού γράφου στην μνήμη).
- Για τους S-F γράφους το πετύχαμε δημιουργώντας αρχικά έναν μικρό S-F γράφο με τις ζητούμενες χαρακτηριστικές ιδιότητες. Στην συνέχεια τον πολλαπλασιάσαμε μέχρι το επιθυμητό μέγεθος και ενώσαμε τους κατάλληλους κόμβους με ακμές, διατηρώντας έτσι τις χαρακτηριστικές του ιδιότητες στον τελικό γράφο.

Το δεύτερο δημιουργεί το αρχείο αλλαγών `edges_to_change.txt`. Αμφότερα αρχεία που δημιουργούνται έχουν αντίστοιχη μορφή με αυτήν των αρχείων που δημιουργεί το `graph_gen.py`.

Με το εργαλείο `graph.cpp` επομένως μπορούμε να κατασκευάσουμε και γράφους με μεγαλύτερα μεγέθη από ό,τι με το εργαλείο `graph_gen.py`, αφού δεν μας περιορίζει σε τόσο μεγάλο βαθμό η μνήμη που έχουμε διαθέσιμη στο υπολογιστικό μας σύστημα. Επιπλέον, παρατηρείται σημαντική βελτίωση στον χρόνο δημιουργίας των αντίστοιχων γράφων.

Υλοποίηση αλγορίθμων γράφων σε Rust με την χρήση timely-dataflow και differential-dataflow

Και οι τρεις αλγόριθμοι που υλοποιήσαμε (triangle counting, pagerank, degree centrality) είναι γνωστοί. Κάνοντας τις κατάλληλες τροποποιήσεις, έτσι ώστε να έχουν αποδοτική υλοποίηση σε Rust και χρησιμοποιώντας παράλληλα κάποια έτοιμα τμήματα κώδικα που υπάρχουν διαθέσιμα στο επίσημο repository του timely-dataflow [8] δημιουργήσαμε τα τρία αρχεία συναρτήσεων `degree_centrality.rs`, `pagerank.rs`, και `triangles.rs`, για τους παραπάνω αλγορίθμους αντίστοιχα. Η καθεμία από τις παραπάνω συναρτήσεις σε Rust δέχεται ως είσοδο δύο αρχεία: το αρχείο `random_graph.txt` με τον αρχικό γράφο, και το αρχείο μεταβολών `edges_to_change.txt` με τις αλλαγές που πρόκειται να πραγματοποιηθούν σε αυτόν.

Ο γράφος φορτώνεται στην μνήμη και ξεκινάει η πρώτη εκτέλεση του αλγορίθμου, όπου γίνεται ο αρχικός υπολογισμός του χρόνου, ο οποίος

εκτυπώνεται στην γραμμή εντολών. Στην συνέχεια η συνάρτηση δρομολογεί τις αλλαγές του γράφου που έχει διαβάσει από το δεύτερο αρχείο, τις πραγματοποιεί και εκτελεί εκ νέου τον αλγόριθμο στον νέο γράφο εκμεταλλευόμενος το υπολογιστικό μοντέλο differential-dataflow. Παράλληλα υπολογίζει τον νέο χρόνο εκτέλεσης τους αλγορίθμου στον νέο γράφο και τον εκτυπώνει στην γραμμή εντολών. Οι χρόνοι αυτοί που εκτυπώθηκαν είναι τα αποτελέσματα που θα χρησιμοποιηθούν για να γίνουν συγκρίσεις και να εξαχθούν τα συμπεράσματα.

Καθορισμός παραμέτρων πειραμάτων

Για να έχουμε όσο το δυνατόν μεγαλύτερο εύρος στα πειράματα μας θα επιλέξουμε κάποιες παραμέτρους, οι οποίες κατά την μεταβολή τους θα επηρεάζουν λογικά τα αποτελέσματα.

Μία πρώτη παράμετρος είναι το είδος του γράφου πάνω στο οποίο θα γίνει η εκτέλεση. Επιλέγουμε ενδεικτικά έναν τυχαίο γράφο (“*random graph”*), όπου κάθε κόμβος συνδέεται με άλλον τυχαία με ορισμένη πιθανότητα P, και έναν γράφο S-F (“*scale-free graph”*), όπου η κατανομή των ακμών ακολουθεί ασυμπτωτικά εκθετική κατανομή με παράμετρο $d = ½P|V|$.

Στην συνέχεια, επιλέγουμε τρία διαφορετικά μεγέθη γράφων, ένα «μικρό», ένα «μεσαίο» και ένα «μεγάλο», τα οποία θα καθοριστούν διαφορετικά για κάθε αλγόριθμο (φαίνονται στους παρακάτω πίνακες), διότι κάθε ένας από αυτούς έχει διαφορετική χρονική και χωρική πολυπλοκότητα. Η μεγαλύτερη τάξη μεγέθους γράφου που μπορεί να επεξεργαστεί το σύστημα μας για κάθε αλγόριθμο θα αποτελέσει τον «μεγάλο» γράφο και ο «μικρός» και ο «μεσαίος» θα υπολογιστούν αναλογικά.

Ως προς τα μεγέθη των γράφων, λαμβάνουμε υπ’ όψιν την μεταβολή του πλήθους των ακμών συναρτήσει του πλήθους των κορυφών, το οποίο είναι αυτό που επηρεάζουμε άμεσα. Ενδεικτικά, βάσει όσων έχουν αναφερθεί για τον καθορισμό των ακμών, οι τρεις random graphs που χρησιμοποιήθηκαν για τα πειράματα με τον αλγόριθμο triangle counting έχουν την εξής σχέση κορυφών και ακμών (όπου $N = |V|$ και $M = |E|$):

$N \rightarrow 2N \rightarrow 5N$ και εφ’ όσον ο πλήρης γράφος έχει $N(N-1)/2$ ακμές,

$M \sim 0.03N^2 \rightarrow M \sim 0.02 \cdot 4N^2 \rightarrow M \sim 0.01 \cdot 25N^2$, δηλαδή $M \rightarrow 3M \rightarrow 8M$.

Μία τρίτη παράμετρος αφορά τις αλλαγές που θα πραγματοποιηθούν στις ακμές του γράφου. Ορίζουμε ως ποσοστό αλλαγών τον λόγο του πλήθους των ακμών που μεταβάλλουμε (προσθέτουμε ή αφαιρούμε) στον γράφο προς το συνολικό πλήθος ακμών του αρχικού γράφου. Τα ποσοστά αλλαγών που επιλέγουμε είναι 1%, 5% και 10% για κάθε γράφο.

Οι ζητούμενοι γράφοι καθώς και τα αρχεία αλλαγών κατασκευάστηκαν με τα εργαλεία `graph_gen.py`, `graph.cpp` και `graph_changes.cpp` που έχουμε παρουσιάσει παραπάνω.

Τέλος, όσον αφορά το βασικό στοιχείο του timely-dataflow που είναι ο καταμερισμός εργασίας σε διάφορα νήματα (worker threads), θα επιλέξουμε τρία διαφορετικά πλήθη από workers. Επιλέγουμε 1, 6 και 12, όπου 12 είναι το βέλτιστο στην προκείμενη περίπτωση, καθώς όπως αναφέραμε το σύστημα πάνω στο οποίο γίνεται η εκτέλεση διαθέτει 12 νήματα. Βέβαια, επειδή ο υπολογιστής διαθέτει 6 πυρήνες, η ουσιαστική βελτίωση αναμένεται να παρατηρηθεί από τον 1 στους 6, ενώ από τους 6 στους 12 αναμένεται να υπάρχουν μικρές διακυμάνσεις, συνήθως προς την κατεύθυνση της βελτίωσης. Αν επιλέγαμε περισσότερους από 12 workers, πιθανώς να παρατηρούσαμε ακόμα και επιδείνωση του χρόνου εκτέλεσης, λόγω περιορισμού του υπολογιστικού συστήματος.

Οι συνδυασμοί παραμέτρων που θα μπορούσαμε δυνητικά να δοκιμάσουμε είναι ατελείωτοι και ξεφεύγουν από τον σκοπό της εργασίας. Για τον λόγο αυτόν επιλέχτηκαν οι $2 \cdot 3 \cdot 3 \cdot 3 = 54$ συνδυασμοί παραμέτρων για κάθε αλγόριθμο, από τους οποίους προκύπτει μία σχετικά ολοκληρωμένη εικόνα για τον σκοπό που εξετάζουμε. Κατασκευάζουμε τρεις πίνακες που συνοψίζουν τις παραπάνω παραμέτρους, έναν για κάθε αλγόριθμο:

Triangle Counting			
Είδος Γράφου	Μέγεθος Γράφου	Ποσοστό Αλλαγών	Workers
Random Graph	10000 nodes (P=3%)	1%	1
	20000 nodes (P=2%)	5%	6
	50000 nodes (P=1%)	10%	12
S-F Graph	10000 nodes (d=50)	1%	1
	10000 nodes (d=150)	5%	6
	20000 nodes (d=200)	10%	12

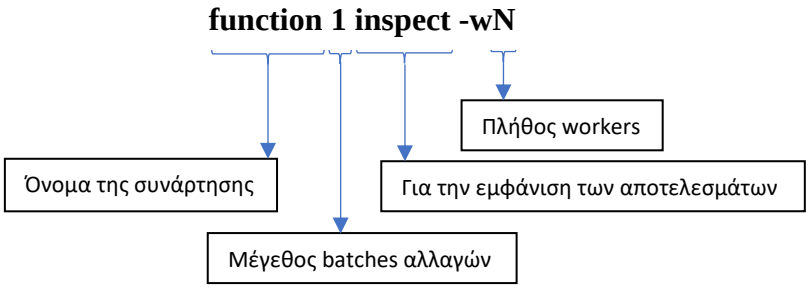
Pagerank			
Είδος Γράφου	Μέγεθος Γράφου	Ποσοστό Αλλαγών	Workers
Random Graph	50000 nodes (P=1%)	1%	1
	200000 nodes (P=1%)	5%	6
	200000 nodes (P=3%)	10%	12
S-F Graph	50000 nodes (d=250)	1%	1
	200000 nodes (d=10^3)	5%	6
	200000 nodes (d=$3 \cdot 10^3$)	10%	12

Degree Centrality			
Είδος Γράφου	Μέγεθος Γράφου	Ποσοστό Αλλαγών	Workers
Random Graph	50000 nodes (P=1%)	1%	1
	150000 nodes (P=3%)	5%	6
	400000 nodes (P=2%)	10%	12
S-F Graph	50000 nodes (d=250)	1%	1
	150000 nodes (d=$2.25 \cdot 10^3$)	5%	6
	400000 nodes (d=$8 \cdot 10^3$)	10%	12

Για τους αλγορίθμους pagerank και degree centrality τα μεγέθη γράφων που χρησιμοποιήθηκαν για τους S-F και random graphs είναι ίδιο. Για τον αλγόριθμο triangle counting, επειδή υπήρχε μεγάλη διαφορά στην ταχύτητα εκτέλεσης μεταξύ S-F και random graph, επιλέχτηκαν διαφορετικά μεγέθη για κάθε είδος γράφου, όπως φαίνεται παραπάνω.

ΕΚΤΕΛΕΣΗ

Η εκτέλεση ενός αλγορίθμου πραγματοποιείται εισάγοντας μία εντολή της μορφής:



Εκτέλεση αλγορίθμου Triangle Counting

Εκτελούμε τον αλγόριθμο triangle counting με την εντολή `triangles 1 inspect`. Αρχικά εμφανίζεται στην γραμμή εντολών το πλήθος των τριγώνων του αρχικού γράφου και ο χρόνος που χρειάστηκε για να υπολογιστούν. Από κάτω εμφανίζεται το πλήθος τριγώνων του νέου γράφου, μετά τις αλλαγές, και ο νέος χρόνος υπολογισμού.

Στιγμιότυπο από την εκτέλεση του αλγορίθμου triangle counting σε S-F γράφο με 10000 κόμβους και d=50, για την οποία έχουμε χρησιμοποιήσει 1 worker (-w1).

Εκτέλεση αλγορίθμου Pagerank

Εκτελούμε τον αλγόριθμο pagerank με την εντολή `pagerank 1 inspect`. Αρχικά εμφανίζεται στην γραμμή εντολών το άθροισμα των τιμών pagerank για κάθε κόμβο του αρχικού γράφου και ο χρόνος που χρειάστηκε για να υπολογιστούν. Από κάτω εμφανίζεται η διαφορά του αθροίσματος των τιμών pagerank του νέου γράφου από την αρχική και ο νέος χρόνος υπολογισμού. Επιλέξαμε να εμφανίσουμε τα αποτελέσματα αθροιστικά και όχι για κάθε κόμβο ξεχωριστά για να είναι πιο ευανάγνωστα.

Στιγμιότυπο από την εκτέλεση του αλγορίθμου triangle counting σε τυχαίο γράφο με 50000 κόμβους και P=1%, για την οποία έχουμε χρησιμοποιήσει 6 workers (-w6).

Εκτέλεση αλγορίθμου Degree Centrality

Εκτελούμε τον αλγόριθμο degree centrality με την εντολή `degree_centrality 1 inspect`. Αρχικά εμφανίζεται στην γραμμή το πλήθος των κόμβων του αρχικού γράφου και ο χρόνος που χρειάστηκε για να υπολογιστούν οι βαθμοί τους (οι οποίοι δεν εμφανίζονται). Από κάτω

εμφανίζεται ο νέος χρόνος υπολογισμού μετά τις αλλαγές που πραγματοποιήθηκαν στον γράφο. Το πλήθος των κόμβων δεν αλλάζει, αφού προστέθηκαν και αφαιρέθηκαν ακμές. Επιλέξαμε να μην εμφανίσουμε τον βαθμό κάθε κόμβου ξεχωριστά για να είναι πιο ευανάγνωστα.

Στιγμιότυπο από την εκτέλεση του αλγορίθμου degree centrality σε τυχαίο γράφο με 50000 κόμβους και P=1%, για την οποία έχουμε χρησιμοποιήσει 6 workers (-w6).

Αυτοματοποίηση των εκτελέσεων με shell script

Καθώς το πλήθος των πειραμάτων που χρειάστηκε να πραγματοποιήσουμε ήταν αρκετά μεγάλο, δημιουργήσαμε ένα shell script το οποίο εισήγαγε και εκτελούσε διαδοχικά τις εντολές, χωρίς να χρειάζεται να περαστούν ξεχωριστά η κάθε μία. Με αυτόν τον τρόπο αυτοματοποιήσαμε την διαδικασία και μειώσαμε τον συνολικό χρόνο διάρκειας των πειραματισμών μας.

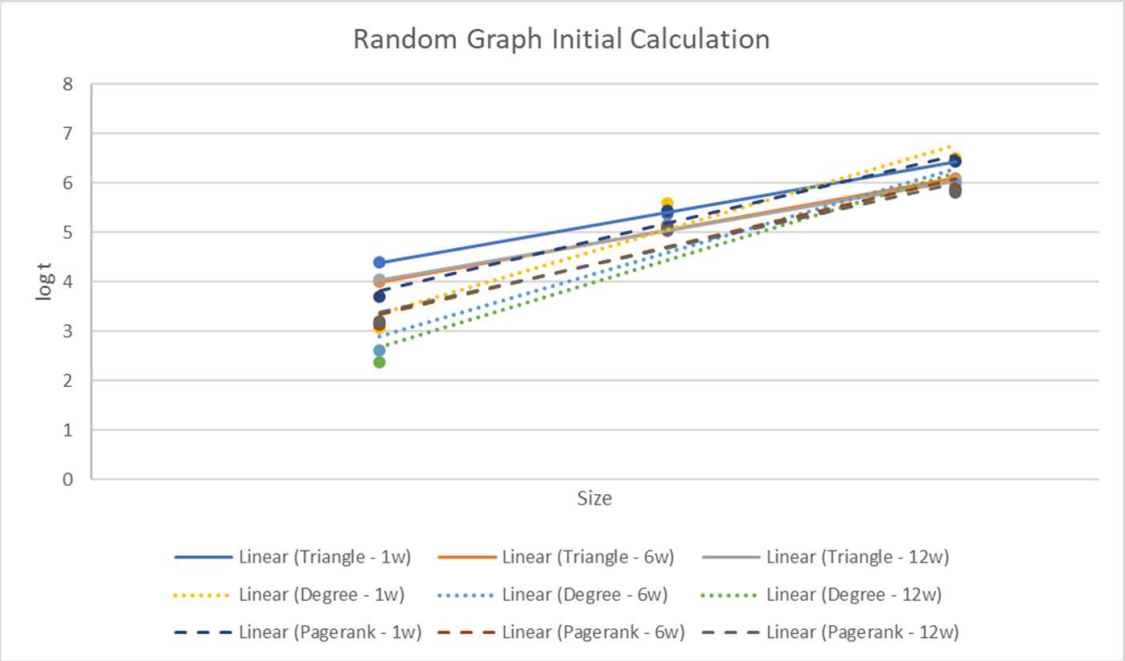
Καταγραφή αποτελεσμάτων

Με την ολοκλήρωση κάθε πειράματος, καταγράφαμε στο συγκεντρωτικό αρχείο, τον αντίστοιχο μετρηθέντα χρόνο πρώτου υπολογισμού και χρόνο επανυπολογισμού μετά τις αλλαγές, σε milliseconds. Ο χρόνος πρώτου υπολογισμού δεν περιελάμβανε τον απαιτούμενο χρόνο για την φόρτωση του αρχείου με τον εκάστοτε γράφο, ο οποίος ήταν σημαντικός, ειδικά για τα αρχεία των μεγάλων γράφων, ούτε τον χρόνο φόρτωσης των αλλαγών σε κάθε πείραμα.

ΑΠΟΤΕΛΕΣΜΑΤΑ

Σχολιασμός χρόνων αρχικού υπολογισμού

Φαίνεται πως το μέγεθος γράφου επηρεάζει με τον αναμενόμενο τρόπο τον απαιτούμενο χρόνο υπολογισμού, καθώς αυτός αυξάνεται για μεγαλύτερο όγκο δεδομένων. Ειδικότερα, ο χρόνος υπολογισμού παρουσιάζει εκθετικής μορφής αύξηση συναρτήσει της αύξησης των μεγεθών (πληθών κορυφών και ακμών) του γράφου. Ανεξαρτήτως πλήθους workers, περισσότερο χρόνο απαιτεί ο υπολογισμός του αλγορίθμου triangles, έπειτα λιγότερο ο pagerank, και τον ελάχιστο στα περισσότερα πειράματα ο degree counting. Με έναν μόνο worker ο χρόνος είναι σημαντικά μεγαλύτερος σε όλες τις διαμορφώσεις, κατά παράγοντα περίπου 2.5, ενώ μεταξύ 6 και 12 workers η διαφορά είναι μικρή, στις περισσότερες μετρήσεις ευνοεί τους 12. Ενδεικτικά, στα πειράματα με τυχαίο γράφο και έξι workers, για τον αλγόριθμο triangle counting, οι χρόνοι είναι της τάξεως των 10^1s για τον μικρό γράφο, 10^2s για τον μεσαίο, και 10^3s για τον μεγάλο. Ομοίως, για τον αλγόριθμο degrees, οι χρόνοι είναι της τάξεως των 10^0s για τον μικρό γράφο, 10^2s για τον μεσαίο, και 10^2s (οριακά 10^3) για τον μεγάλο. Τέλος, για τον pagerank με τις ίδιες ρυθμίσεις, οι χρόνοι είναι της τάξεως των 10^0s για τον μικρό γράφο, 10^2s για τον μεσαίο, και 10^2s για τον μεγάλο.



Στο διάγραμμα αναπαρίσταται η σχέση μεταξύ μεγέθους γράφου και λογαρίθμου χρόνου πρώτου υπολογισμού, σε διαφορετικά σύνολα ανά πλήθος workers (διαφοροποίηση χρώματος) και ανά αλγόριθμο (τύπος γραμμής). Στον χρόνο εφαρμόζεται λογάριθμος ώστε να αποτυπωθεί καλύτερα η τάση αύξησης αυτού. Χαράσσοντας ευθείες βέλτιστων τετραγώνων παρατηρείται αυξητική συμπεριφορά του χρόνου παρεμφερής με εκθετική, όπως ήταν αναμενόμενο βάσει των αυξήσεων των παραμέτρων του γράφου. Το ανωτέρω περιέχει ενδεικτικά τις μετρήσεις για τον τυχαίο γράφο. Οι ίδιες ακριβώς έγιναν και για τον scale-free.

Στο εξής σχολιάζουμε λαμβάνοντας υπ' όψιν τους χρόνους επανυπολογισμού. Είναι σημαντικό να επισημανθεί ότι, δίχως την χρήση του συστήματος differential dataflow, με συμβατικές μεθόδους, οι χρόνοι αυτοί θα ταυτίζονταν πρακτικώς με τους αρχικούς, καθώς όλοι οι υπολογισμοί θα γίνονταν εκ νέου.

Σχολιασμός ως προς πλήθος workers

Η επιλογή των τριών αυτών πληθών workers έγινε αφ' ενός για να εξετασθεί η υποθετική συμπεριφορά του differential σε single-thread επεξεργαστικό σύστημα, και αφ' ετέρου διότι ο υπολογιστής στον οποίον εκτελέσθηκαν τα πειράματα διέθετε εξαπύρηνο επεξεργαστή με 12 threads. Οπότε εξετάσθηκε η συμπεριφορά για έναν worker, για πλήθος workers ισάριθμο με τους πυρήνες, και για πλήθος ισάριθμο με τα threads. Όπως παρατηρήθηκε για τους χρόνους αρχικού υπολογισμού, και στους χρόνους επανυπολογισμού η σημαντικότερη διαφορά υπάρχει μεταξύ 1 και 6 workers, ενώ μεταξύ 6 και 12 η διαφορά είναι μικρότερη, ως αναμενόμενο. Αναλυτικά, ο απαιτούμενος χρόνος για 1 worker είναι περίπου τριπλάσιος από αυτόν για 6, ενώ ο χρόνος για 12 στις περισσότερες μετρήσεις κυμαίνεται γύρω από το 90% του χρόνου των 6, όμως σε κάποιες περιπτώσεις είναι μεγαλύτερος. Συγκεκριμένα, αυτό παρατηρείται σε μικρότερους γράφους ή μικρά ποσοστά αλλαγών, όπου το χρονικό κόστος συγχρονισμού των workers υπερβαίνει την βελτίωση που η αύξηση αυτών επιφέρει. Συνεπώς, είναι εμφανές πως η εκμετάλλευση των πολλαπλών workers συνήθως οδηγεί σε σημαντική βελτίωση της απόδοσης, η οποία είναι ήδη καλή και με έναν, χάρη στο timely. Από 6 σε 12 παρατηρείται φθίνουσα επιστροφή.

Σχολιασμός ως προς μέγεθος αλλαγών και μέγεθος γράφου

Εξετάζοντας χωριστά τις εννέα μετρήσεις που αναλογούν σε κάθε worker, παρατηρείται σταθερή αύξηση του απόλυτου χρόνου επανυπολογισμού, με τον χρόνο για το μεγάλο γράφημα και μεγάλο ποσοστό αλλαγών να είναι κατά δύο τάξεις μεγέθους μεγαλύτερος από αυτόν για το μικρό γράφο και μικρό ποσοστό αλλαγών. Το ίδιο δεν παρατηρείται όμως και για την αναλογία μεταξύ χρόνου αρχικού υπολογισμού και χρόνου δεύτερου υπολογισμού ύστερα από τις αλλαγές. Εκεί τα αποτελέσματα διαφέρουν ανά αλγόριθμο. Στον triangles παρατηρείται περιοδική αυξομείωση σε κάθε κύκλο αλλαγής μεγέθους γράφου, ενώ στους άλλους δύο η αναλογία είναι μικρότερη στο μεγαλύτερο γράφημα, χωρίς όμως κάποια σαφής σχέση να μπορεί να παρατηρηθεί. Φαίνεται πως το σύστημα αποδίδει ακόμη καλύτερα σε μικρότερα μεγέθη. Εν πάση περιπτώσει βέβαια η βελτίωση είναι εκπληκτική.

Σχολιασμός ως προς είδος γράφου

Ως προς το είδος του γράφου, φαίνεται πως οι χρόνοι εκτέλεσης ήταν μεγαλύτεροι στον τυχαίο για τους αλγορίθμους degree και pagerank, ενώ στον triangles απαίτησε περισσότερο ο scale-free. Στους δύο πρώτους κατά σειρά αλγορίθμους, triangle counting και degree centralities, η διαφορά είναι σημαντική, ενώ στον pagerank αμελητέα. Ειδικά για τον αλγόριθμο triangle counting, η εν λόγω διαφορά στους χρόνος οφείλεται κυρίως στο γεγονός πως, για τα πειράματα επί της μορφολογίας scale-free, χρησιμοποιήθηκαν γράφοι μικρότερων διαστάσεων, λόγω υπολογιστικών περιορισμών όπως προαναφέρθηκε. Πρέπει να σημειωθεί ότι οι μετρήσεις μεταξύ των δύο ειδών γράφων διαφέρουν μόνο ως προς τους απόλυτους χρόνους εκτέλεσης και προσαρμογής αποτελέσματος, και όχι ως προς την αναλογία μεταξύ αυτών των δύο, η οποία παραμένει σχετικά σταθερή.

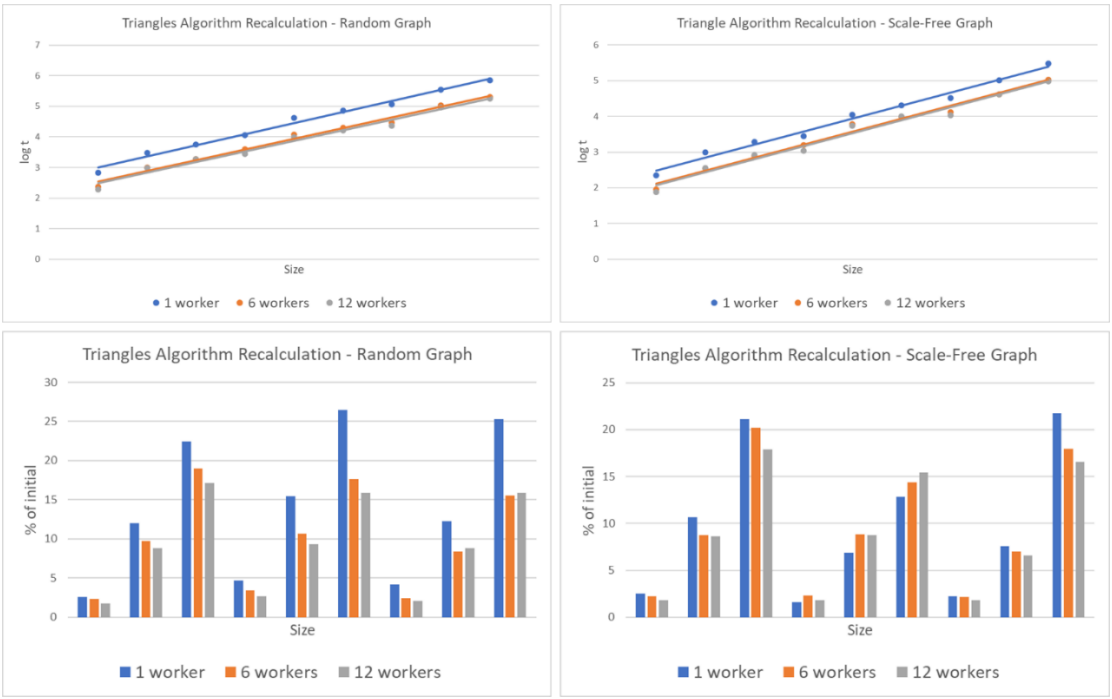
Σχολιασμός ως προς αλγόριθμο

Οι μετρήσεις για τους τρεις αλγορίθμους παρουσιάζουν αξιοσημείωτες διαφορές. Ο αλγόριθμος degree είναι αυτός που απαιτεί λιγότερο χρόνο υπολογισμού. Ενδεικτικά, οι τιμές αυτού για 6 workers να βρίσκονται στο διάστημα 150-1000ms. Ο ίδιος οδηγεί και σε καλύτερη απόδοση του timely, με ποσοστά 0.1-20% κατά μήκος όλων των μεγεθών γράφων και αλλαγών. Ο triangle counting απαιτεί αρχικό χρόνο υπολογισμού μεταξύ

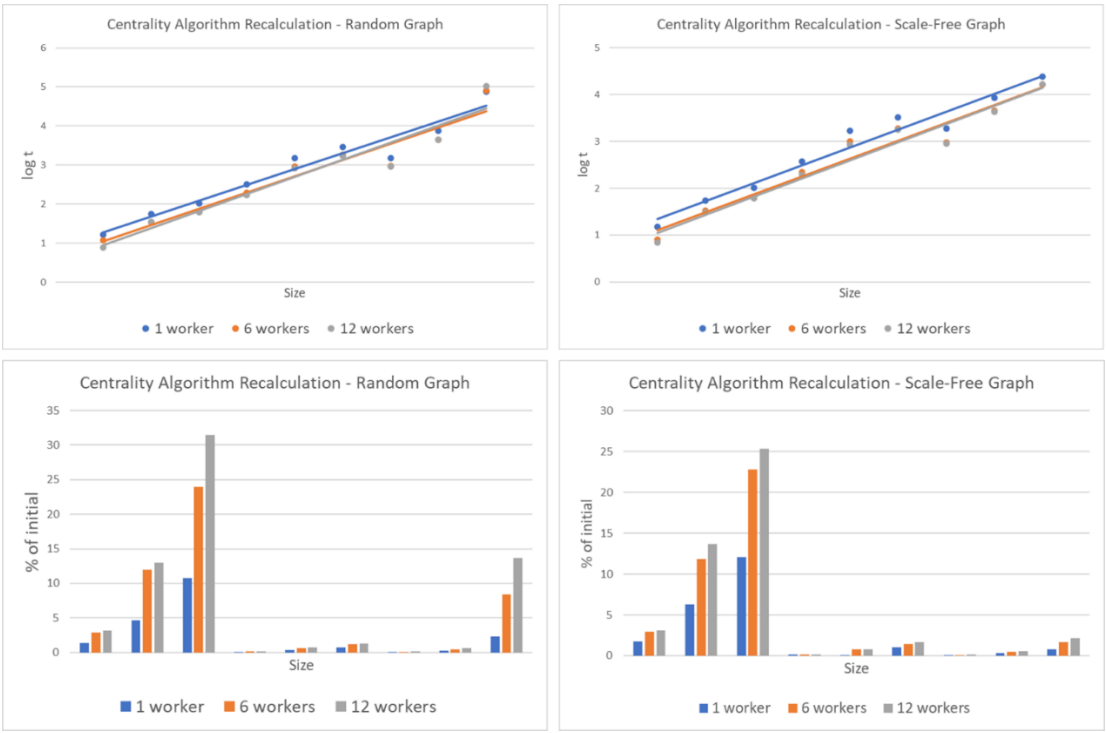
10 και 1200s, πάλι ενδεικτικά για 6 workers και συνυπολογίζοντας όλα τα πειράματα, ενώ ο χρόνος επανυπολογισμού είναι μεταξύ 2 και 20% του αρχικού. Τέλος, ο pagerank απαιτεί αρχικό χρόνο υπολογισμού μεταξύ 1 και 800s, πάλι για 6 workers και συνυπολογίζοντας όλα τα πειράματα, ενώ ο χρόνος επανυπολογισμού είναι μεταξύ 10 και 60% του αρχικού. Εξετάζοντας τις αντίστοιχες μετρήσεις για τα άλλα πλήθη workers παρατηρείται η ίδια συμπεριφορά. Επομένως, φαίνεται ότι το timely dataflow αποδίδει πολύ καλύτερα για τους δύο πρώτους, και ειδικά για τον degree, όπου η απόδοση ειδικά για τα μεσαία και μεγάλα γραφήματα αμφοτέρων ειδών είναι εντυπωσιακή. Μεταξύ των αλγορίθμων νόημα έχει η σύγκριση μόνο της αναλογίας χρόνου επανυπολογισμού προς αρχικού, καθώς δεν χρησιμοποιήθηκαν τα ίδια μεγέθη γράφων σε όλους. Ο pagerank, καθώς πρόκειται για πιο σύνθετο υπολογισμό, φαίνεται λογικό να οδηγεί σε χαμηλότερη απόδοση του συστήματος dataflow.

Συνολικό σχόλιο για ποσοστό

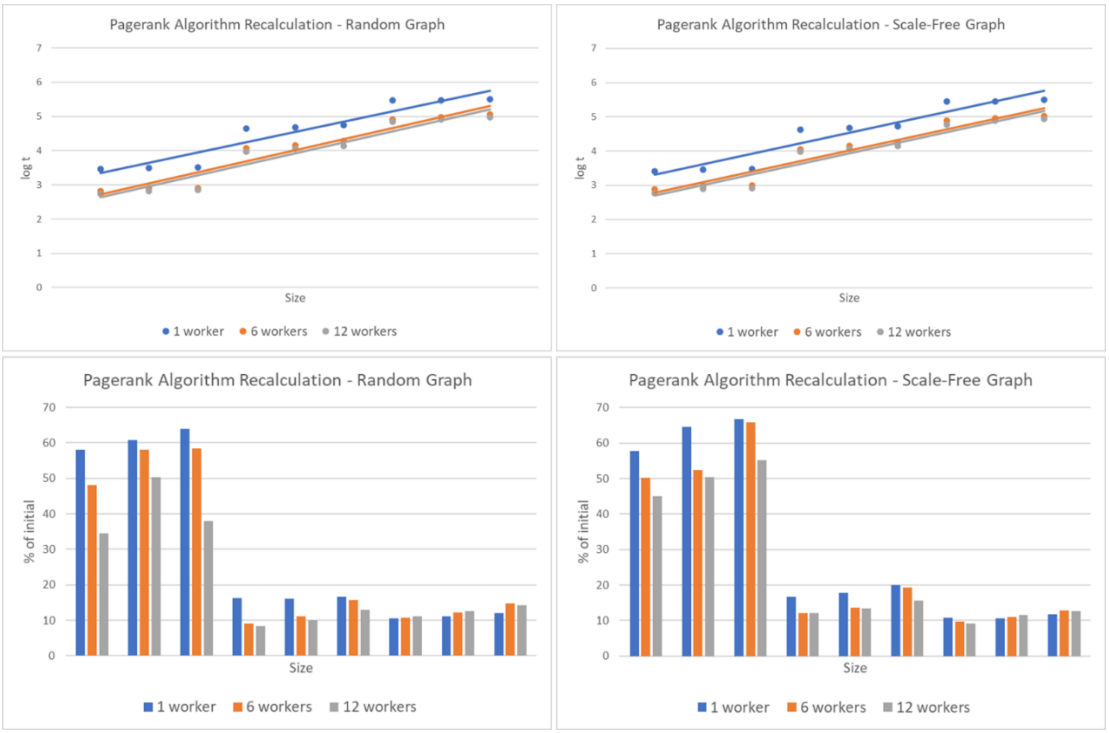
Η επιπρόσθετη στον πίνακα στήλη όπου αναγράφεται σε εκατοστιαίο ποσοστό ο λόγος του χρόνου επανυπολογισμού προς τον αρχικό δίνει σημαντικότατη πληροφορία για την απόδοση του συστήματος differential dataflow. Μελετώντας την μεταβολή αυτήν γίνεται εμφανής η βελτίωση που επιφέρει το timely. Ενώ με συμβατικούς υπολογισμούς κάθε αλλαγή σε γράφο θα απαιτούσε χρόνο υπολογισμού ίσο με τον αρχικό, το differential κάνει τους υπολογισμούς σε χρόνο έως και 1000 φορές λιγότερο. Ακόμη και στις χειρότερες περιπτώσεις, δεν υπερβαίνει το 60% του αρχικού, η χρήση του δηλαδή είναι πάντοτε ωφέλιμη.



Γραφική αναπαράσταση των αποτελεσμάτων των πειραμάτων για τον αλγόριθμο triangle counting, σχετικά με τους χρόνους επανυπολογισμού, για τα δύο είδη γράφων. Στο πρώτο διάγραμμα για κάθε είδος αναπαρίσταται ο λογάριθμος απολύτου χρόνου επανυπολογισμού συναρτήσει μεγέθους πειράματος. Ως μέγεθος πειράματος στον οριζόντιο άξονα στο εξής θα αναφέρονται οι 18 μετρήσεις που έχουν πραγματοποιηθεί για κάθε συνδυασμό είδους γράφου και μεγέθους ποσοστού αλλαγών, με την λογική σειρά που παρουσιάζονται και στον πίνακα αποτελεσμάτων. Πρώτο σημείο, δηλαδή, μικρός γράφος με μικρό ποσοστό αλλαγών, ύστερα μικρός γράφος με μεσαίο ποσοστό αλλαγών, και ούτω καθ' εξής. Μεταξύ αυτών παρατηρείται ισχυρή γραμμική σχέση, επομένως ο χρόνος αυξάνεται εκθετικά. Ωστόσο, μελετώντας τον λόγο χρόνου επανυπολογισμού προς χρόνο πρώτου υπολογισμού, ο οποίος δίνει και καλύτερη εικόνα για την απόδοση του εξεταζόμενου συστήματος, παρατηρείται περισσότερο σύνθετη συμπεριφορά. Συγκεκριμένα, όπως φαίνεται στα δεξιά διαγράμματα, ειδικά για τον αλγόριθμο αυτόν φαίνεται πως ο λόγος αυτός είναι σχετικά χαμηλότερος για μικρά ποσοστά αλλαγών, ανεξαρτήτως μεγέθους γραφήματος, και υψηλότερος (άρα χαμηλότερη η απόδοση) για τα μεγάλα ποσοστά αλλαγών.



Γραφική αναπαράσταση των αποτελεσμάτων των πειραμάτων για τον αλγόριθμο μέτρησης των degree centralities, σχετικά με τους χρόνους επανυπολογισμού, για τα δύο είδη γράφων. Παρ' ότι και σε αυτήν την περίπτωση παρατηρούνται όμοια αποτελέσματα στον απόλυτο χρόνο επανυπολογισμού ως προς τον τρόπο εκθετικής αύξησης, τα ποσοστά των αρχικών χρόνων διαφέρουν αρκετά. Εν προκειμένω προέκυψαν σημαντικά υψηλότερα στα τρία πρώτα πειράματα, δηλαδή με τον μικρό γράφο, και στην συνέχεια ήταν χαμηλότερα για τα υπόλοιπα.



Γραφική αναπαράσταση των αποτελεσμάτων των πειραμάτων για τον αλγόριθμο pagerank, σχετικά με τους χρόνους επανυπολογισμού, για τα δύο είδη γράφων. Και πάλι εμφανίζεται γραμμική συσχέτιση στο διάγραμμα λογαρίθμου χρόνου επανυπολογισμού συναρτήσει μεγέθους πειράματος, και ως προς το ποσοστό φαίνεται και πάλι πως αυτό είναι υψηλότερο στα τρία πειράματα με τον μικρό γράφο, για όλα τα διαφορετικά πλήθη workers. Όπως και στα προηγούμενα πειράματα, οι κυανές τιμές, που αντιστοιχούν στις μετρήσεις για έναν worker είναι ορατά υψηλότερες (δηλαδή η απόδοση χειρότερη), ενώ για τους 6 και τους 12 οι τιμές είναι πλησιέστερες μεταξύ τους.

Επεκτάσεις

Η κυριότερη και πλέον ενδιαφέρουσα επέκταση για την παρούσα εργασία θα ήταν η εκτέλεση παρομοίων πειραμάτων με πολύ μεγαλύτερους γράφους, που να προσομοιώνουν τους πραγματικούς που παράγονται από τα σύγχρονα κοινωνικά δίκτυα. Αυτό θα ήταν εφικτό μόνον με την χρήση συστήματος κατανεμημένης επεξεργασίας, όπως το Apache Spark. Ως προς την υλοποίηση, η γλώσσα Rust φαίνεται πως είναι κατάλληλη, λόγω ασφάλειας και ταχύτητας, υποδομή για το Naiad. Ωστόσο, η σχετική βιβλιοθήκη υστερεί σημαντικά τόσο σε τεκμηρίωση όσο και σε ευχρηστία, συνεπώς η προγραμματιστική διεπαφή του timely dataflow πρέπει να βελτιωθεί. Τέλος, για την επικύρωση των αποτελεσμάτων που εξαγάγαμε, θα μπορούσαμε να επαναλάβουμε τα πειράματα, το οποίο στο πλαίσιο της παρούσης εργασίας ήταν χρονικώς ανέφικτο.

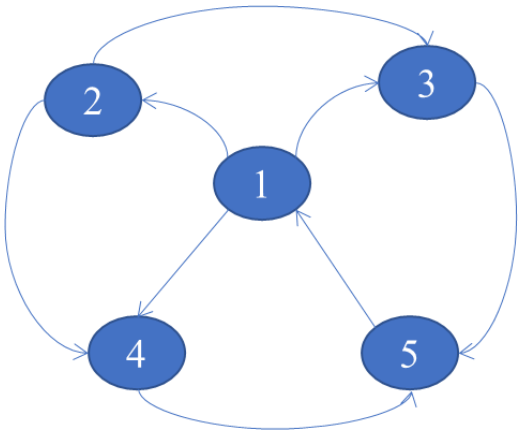
ΕΠΙΛΟΓΟΣ

Τα ευρήματα των πειραματισμών της παρούσης εργασίας επιβεβαιώνουν πως τα υπολογιστικά μοντέλα Timely Dataflow και Differential Dataflow είναι αποδοτικά σε βαθμό μεγαλοφυή. Μπορούν να βρουν εξαιρετική εφαρμογή σε κλάδους της ανάλυσης μεγάλων δεδομένων που απαιτούν ταχεία ενημέρωση αποτελεσμάτων σε πραγματικό χρόνο.

ΠΑΡΑΡΤΗΜΑ

Ο αλγόριθμος Pagerank

Ο αλγόριθμος pagerank εφαρμόζεται κυρίως σε κατευθυνόμενα γραφήματα. Στην απλούστερη μορφή του, χρησιμοποιεί την αναπαράσταση του γραφήματος ως πίνακα πιθανοτήτων μεταβάσεων (A), με όλες τις μεταβάσεις ισοπίθανες (δηλαδή αν μία κορυφή έχει n εξερχόμενες ακμές, το στοιχείο ανά γραμμή κάθε μίας εξ αυτών στην στήλη του πίνακα για την κορυφή αυτήν έχει τιμή 1/n και οι υπόλοιπες 0). Η κατάταξη των κορυφών προκύπτει από την εύρεση του ιδιοδιανύσματος του πίνακα μεταβάσεων που αντιστοιχεί στην ιδιοτιμή 1. Η ύπαρξη της ιδιοτιμής αυτής είναι εγγυημένη καθώς κάθε στήλη του A έχει άθροισμα 1, άρα κάθε στήλη του A-I έχει άθροισμα 0, οπότε η ορίζουσα του πίνακα αυτού είναι μηδενική.



Πίνακας πιθανοτήτων μεταβάσεων:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1/3 & 0 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$(A - I)x = 0 \Rightarrow \begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ 1/3 & -1 & 0 & 0 & 0 \\ 1/3 & 1/2 & -1 & 0 & 0 \\ 1/3 & 1/2 & 0 & -1 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = 0$$

$$-x_1 + x_5 = 0$$

$$x_1/3 - x_2 = 0$$

$$x_1/3 + x_2/2 - x_3 = 0$$

$$x_1/3 + x_2/2 - x_4 = 0$$

$$x_3 + x_4 - x_5 = 0$$

Από το παραπάνω σύστημα προκύπτει ότι ιδιοδιάνυσμα για $\lambda = 1$ είναι οποιοδήποτε πολλαπλάσιο του (1, 1/3, 1/2, 1/2, 1). Αυτή είναι και η κατάταξη των κορυφών σύμφωνα με τον pagerank, δηλαδή πρώτες οι 1 και 5, ύστερα οι 3 και 4, και τελευταία η 2. Η κατάταξη αυτή υποδεικνύει διαισθητικά πόσο «δημοφιλείς» είναι οι κορυφές. Σε μία πραγματική εφαρμογή οι κορυφές του γραφήματος θα μπορούσαν να αναπαριστούν ιστοσελίδες, και οι ακμές υπερ-συνδέσμους προς άλλες σελίδες. (Η «πραγματική εφαρμογή» αυτή είναι κοινώς γνωστή ως google).

Για τον υπολογισμό του pagerank σε μεγάλα γραφήματα με υπολογιστική μέθοδο χρησιμοποιείται η μέθοδος των «τυχαίου περιηγητή» (“random surfer”), η προσομοίωση, δηλαδή, τυχαίας διαδρομής στο γράφημα. Με αρκετές επαναλήψεις, η συχνότητα προσγείωσης σε κάθε κορυφή προσεγγίζει την πραγματική κατάταξη. Προκειμένου να αντιμετωπίζονται προβλήματα που προκαλούνται από την μορφολογία του γραφήματος, όπως αδιέξοδα, δηλαδή κορυφές χωρίς εξερχόμενες ακμές, ή μη συνδεόμενες συνιστώσες του γραφήματος, εισάγεται ένας συντελεστής απόσβεσης (“damping factor”). Αυτός συνήθως λαμβάνει τιμή περίπου 85% και υποδηλώνει ότι η επόμενη μετάβαση της προσομοίωσης προκύπτει από τις δυνατές στην παρούσα κατάσταση κατά αυτήν την πιθανότητα, ή είναι εντελώς τυχαία κατά το συμπλήρωμα αυτής (εν προκειμένω 15%).

ΒΙΒΛΙΟΓΡΑΦΙΑ

Έντυπη

[1] Anuradha, J. "A Brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology". *Procedia computer science vol. 48*, 2015, p. 319-324.

[2] Blandy, Jim, et al. *Programming Rust*. Sebastopol: O'Reilly, 2021. Print.

[3] Cormen, Thomas H., et al. *Introduction to Algorithms*. 3rd ed., Cambridge, MIT Press, 2009.

[4] Sedgewick, Robert. *Algorithms in C++ Part 5: Graph Algorithms*, 3rd ed. Indianapolis: Addison-Wesley, 2002. Print.

[5] Silberschatz, Abraham, κ.α. *Συστήματα Βάσεων Δεδομένων*, 6η έκδ. Μετεφρασμένο από την Μαίρη Γκλαβά. Αθήνα: Εκδόσεις Γκιούρδας, 2018.

Ηλεκτρονική

[6] "Degree Centrality". *Neo4j Graph Data Science*. Neo4j Inc, 6 Sep. 2022, https://neo4j.com/docs/graph-data-science/2.1/algorithms/degree-centrality/.

[7] McSherry, Frank. "differential-dataflow". *GitHub*, GitHub Inc., Aug. 2019, https://github.com/frankmcsherry/differential-dataflow.

[8] McSherry, Frank. "timely-dataflow". *GitHub*, GitHub Inc., Aug. 2019, https://github.com/frankmcsherry/timely-dataflow.

[9] McSherry, Frank, et al. “Naiad: a Timely Dataflow System”. *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, Nov. 2013, p. 439-455. https://doi.org/10.1145/2517349.2522738.

[10] McSherry, Frank, et al. “Differential Dataflow”. C*onference on Innovative Data Systems Research CIDR*, 2013.

[11] “Pagerank”. *Neo4j Graph Data Science*. Neo4j Inc, 6 Sep. 2022, https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/.

[12] “Reference”. *NetworkX Network Analysis in Python*. NetworkX Developers, 14 Jun. 2022, https://networkx.org/documentation/stable/reference/index.html.

[13] “Triangle Count”. *Neo4j Graph Data Science*. Neo4j Inc, 6 Sep. 2022, https://neo4j.com/docs/graph-data-science/current/algorithms/triangle-count/.