Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Σύγκριση συστημάτων επεξεργασίας γράφων μεγάλης κλίμακας σε περιβάλλον πολλαπλών συστημάτων

Διπλωματικη Εργασια

του

ΝΙΚΟΛΑΟΥ ΦΩΤΟΥ

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2018

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Σύγκριση συστημάτων επεξεργασίας γράφων μεγάλης κλίμακας σε περιβάλλον πολλαπλών συστημάτων

Διπλωματικη Εργασια

του

ΝΙΚΟΛΑΟΥ ΦΩΤΟΥ

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 13η Νοεμβρίου 2018.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Νεκτάριος Κοζύρης	Γιώργος Γκούμας	Δημήτριος Τσουμάκος
Καθηγητής	Καθηγητής	Αναπληρωτής Καθηγητής
Ε.Μ.Π	Ε.Μ.Π	Πανεπιστήμιο Ιονίου

Εργαστηριο Υπολογιστικων Συστηματων
Αθήνα, Νοέμβριος 2018

(Υπογραφή)

.....

Φωτος Νικολαος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Η αναπαράσταση και επεξεργασία μεγάλου όγκου δεδομένων σε μορφή γράφου είναι απαραίτητη για μεγάλο πλήθος εφαρμογών. Προς αυτή την κατεύθυνση, βιομηχανία και ακαδημαϊκή κοινότητα προχώρησαν στη δημιουργία ποικίλων συστημάτων επεξεργασίας γράφων. Ωστόσο, οι σύγχρονες ροές εργασιών αποτελούνται από εργασίες που χρειάζονται περισσότερες από μία πλατφόρμες προκειμένου να αποδώσουν βέλτιστα.

Σκοπός, λοιπόν, της διπλωματικής εργασίας είναι η μελέτη ενός περιβάλλοντος με πολλά συστήματα επεξεργασίας γράφων σε κατανεμημένο περιβάλλον. Συγκεκριμένα, θα μελετήσουμε τέσσερα από τα πιο διαδεδωμένα συστήματα επεξεργασίας γράφων πάνω στα οποία θα υλοποιήσουμε τρεις γνωστούς αλγορίθμους γράφων. Ύστερα, θα συγκρίνουμε την επίδοση των συστημάτων πάνω σε αυτούς τους αλγορίθμους βάσει κριτηρίων που θα ορίσουμε. Τέλος, με τη βοήθεια του μετα-δρομολογητή IReS θα δείξουμε ότι για δεδομένη ροή εργασιών υπάρχει συνδυασμός των συστημάτων επεξεργασίας που να πετυχαίνει καλύτερες επιδοσεις έναντι της εκτέλεσής της από κάθε σύστημα ξεχωριστά.

Λέξεις Κλειδιά

Επεξεργασία γράφων, Κατανεμημένη επεξεργασία, Μεγάλα Δεδομένα, Παράλληλη Επεξεργασία.

Abstract

Large-scale graph data representation and processing is necessary to various applications. To that extent, both industry and academia produced a variety of graph processing platforms. However, in order for work-flows to be executed optimally, more than one platforms are required.

The purpose of this diploma is to study a multi-engine graph-processing environment. In particular, we study four state-of-the-art graph-processing engines and implement three graph algorithms on them. Secondly, we compare the performance of every platform based on user-defined criteria. Finally, based on information extracted from meta-scheduler IReS, we show that for a fixed workflow a combination of platforms can achieve better results than if we executed the workflow with every platform separately.

Keywords

Big Data, Parallel Processing, Graph Processing, Distributed Processing.

Ευχαριστίες

Προτίστως, θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Νεκτάριο Κοζύρη για την ευκαρία που μου δόθηκε να εκπονήσω τη διπλωματική εργασία στο Εργαστήριο Υπολογιστικών Συστημάτων.

Κατόπιν, θα ήθελα να ευχαριστήσω την κ. Αικατερίνη Δόκα που μου έδωσε τη δυνατότητα να ασχοληθώ με αυτό το θέμα καθώς και για την καθοδήγηση που μου παρείχε καθόλη τη διάρκεια της διπλωματικής εργασίας.

Επίσης θα ήθελα να ευχαριστήσω όλη το προσωπικό του εργαστηρίου υπολογιστικών συστημάτων που με βοήθησαν όποτε χρειαζόμουν βοήθεια σχετικά με το στήσιμο των πειραμάτων. Θα ήθελα να ευχαριστήσω ιδιαίτερα τον κ. Γιάγκο Μυτιλίνη για τη καταλυτική συνεισφορά του στην προσπάθειά μου να κατανοήσω την πλατφόρμα IReS.

Τέλος, ευχαριστώ τους γονείς μου για την αγάπη και τη στήριξη που μου έδειχναν όλα αυτά τα χρόνια γιατί χωρίς αυτούς δε θα μπορούσα να φτάσω ως εδώ.

Περιεχόμενα

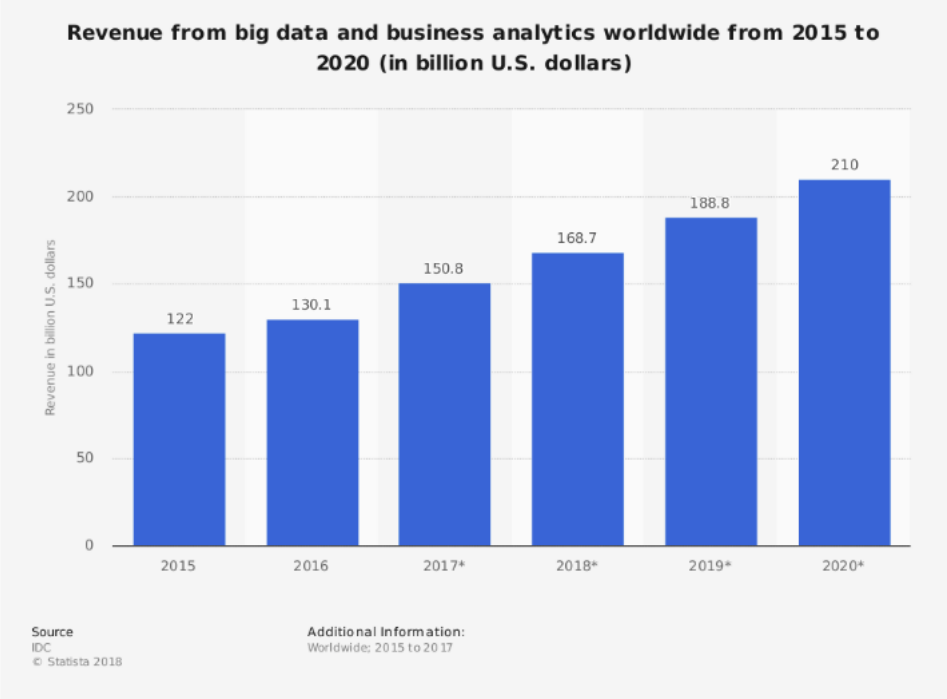
Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Η αποδοτική επεξεργασία και ανάλυση των δεδομένων σε μεγάλη κλίκαμα αποτελεί μείζον ζήτημα τόσο στη βιομηχανία όσο και στον ακαδημαϊκό χώρο. Συγκεκριμένα, πάνω από δύο πεντάκις εκατομμύρια bytes δεδομένων παράγονται καθημερινά [1] και η επεξεργασία τους καθίσταται ζωτικής σημασίας με εφαρμογές να εκτείνονται από τον επιχειρισιακό τομέα και την ανάλυση τάσεων και ανθρώπινης συμπεριφοράς μέχρι τον επιστημονικό τομέα και την ανάλυση βιολογικών δομών και γονιδιωματικής. Με την πάροδο του χρόνου αυξάνεται επίσης και η πολυπλοκότητα των σχέσεων και της αλληλοεξάρτησης των δεδομένων. Ως αποτέλεσμα τα δεδομένα δεν αρκεί να αναπαρίστανται ως ένα σύνολο από στήλες με τιμές αλλά πρέπει να εκφραστούν βέλτιστα και οι σχέσεις μεταξύ τους.



Σχήμα 1.1: Η διαρκώς αυξανόμενη ανάγκη για ανάλυση μεγάλου όγκου δεδομένων

Ο γράφος αποτελεί έναν εναλλακτικό, πιο αποδοτικό, τρόπο μοντελοποίησης δεδομένων

που παρουσιάζουν τα παραπάνω χαρακτηριστικά. Όσο περνάει ο καιρός, ολοένα και περισσότερες εφαρμογές επεξεργάζονται μεγάλο όγκο δεδομένων σε μορφή γράφου σε τομείς όπως η αστρονομία, οι κοινωνικές επιστήμες, οι τηλεπικοινωνίες, η βιολογία. Αυτή η τάση έχει οδηγήσει σε γράφους με εκατομμύρια, ακόμη και τρισεκατομμύρια κόμβους και ακμές [2] με αποτέλεσμα να καθίσταται απαραίτητη η εύρεση εργαλείων που να στοχεύουν στην αποδοτική επεξεργασία τέτοιων δομών.

Το μεγάλο εύρος των εφαρμογών που αξιοποίησαν την ανάλυση γράφων οδήγησε στην δημιουργία ενός μεγάλου πλήθους συστημάτων, γενικού και ειδικού σκοπού. Η κλιμακώσιμη επεξεργασία μεγάλων γράφων αποτελεί πρόκληση αφενός λόγω της δομής του γράφου, αφετέρου λόγω της επαναληπτικής φύσης των αλγορίθμων ανάλυσης γράφων. Βασική ένδειξη της δυσκολίας στην επεξεργασία γράφων αποτελεί η αρχική προσπάθεια για ανάλυση γράφων μεγάλης κλίμακας με το προγραμματιστικό μοντέλο Map Reduce [3], το κατεξοχήν δημοφιλές προγραμματιστικό πλαίσιο για επεξεργασία μεγάλου όγκου δεδομένων. Τα αποτελέσματα δεν ήταν ενθαρρυντικά λόγω των περιορισμών που έθετε το προγραμματιστικό μοντέλο στους αλγορίθμους γράφων [4, 5] που οδηγούσαν σε ραγδαία πτώση της απόδοσης του πλαισίου. Έπειτα από το εγχείρημα του Map Reduce παρουσιάστηκε πληθώρα προγραμματιστικών μοντέλων και συστημάτων τα οποία σχεδιάστηκαν για ανάλυση γράφων όπως το Apache Spark GraphX [20], Apache Flink Gelly [21], Giraph [22], Pregel [26], Neo4j [23], GraphChi [18] και πολλά άλλα.

Καθένα από τα συστήματα αυτά καλύπτει συγκεκριμένες ανάγκες των εφαρμογών, ωστόσο δεν έχει βρεθεί το σύστημα που να υπερτερεί καθολικά έναντι των υπολοίπων. Μερικές από τις απαιτήσεις μίας σύγχρονης εφαρμογής είναι η εκφραστικότητα της μοντελοποίησης του γράφου, η αποδοτική υποβολή ερωτημάτων και επεξεργασίας, η υψηλή απόδοση και κλιμακωσιμότητα, η υποστήριξη δοσοληψιών, η προγραμματιστική ευκολία, η δυνατότητα οπτικοποίησης των γράφων. Οι απαιτήσεις είναι πολλές και δεν υπάρχει μία πλατφόρμα επεξεργασίας γράφων που να τις αντιμετωπίζει βέλτιστα όλες.

Έχουν γίνει αρκετές έρευνες στις οποίες γίνεται προσπάθεια να συγκριθούν οι πλατφόρμες επεξεργασίας γράφων πάνω σε εφαρμογή που δίνει ο χρήστης [6, 7, 8, 9]. Τα αποτελέσματα ποικίλουν ενώ, όπως αναφέραμε, δεν υπάρχει κάποιο σύστημα που επικρατεί πάντα έναντι των υπολοίπων.

Παράλληλα, οι σύγχρονες εφαρμογές απαιτούν ολοένα και πιο πολύπλοκες ροές επεξεργασίας δεδομένων με την ύπαρξη περισσότερων τελεστών και διαφορετικής μορφής αναπαράστασης δεδομένων. Τα δύο παραπάνω σε συνδυασμό με την απαίτηση για αποδοτική επεξεργασία δεδομένων σε μεγάλη κλίμακα οδήγησαν στα περιβάλλοντα πολλαπλών συστημάτων [11]. Εκεί ο χρήστης θα μπορεί να εκτελέσει την εφαρμογή που επιθυμεί γνωρίζοντας ότι για κάθε στάδιό της μπορεί να χρησιμοποιήσει τη βέλτιστη πλατφόρμα επεξεργασίας.

Στα πλαίσια του πολυσυστηματικού περιβάλλοντος προτάθηκε η έννοια ενός συστήματος το οποίο είναι σε θέση να δρομολογεί τα στάδια της εργασίας καθώς επίσης και να επιλέγει βάσει κριτηρίων ποια από τις διαθέσιμες πλατφόρμες είναι η βέλτιστη για επεξεργασία [12]. Ο μετα-δρομολογητής που πρόκειται να μελετήσουμε καλείται IReS. Μέχρι στιγμής, το σύστημα IReS δεν έχει μελετηθεί σε αναλυτική πάνω σε δεδομένα γράφων. Θα προσπαθήσουμε,

λοιπόν, να μετρήσουμε την επίδοση του IReS πάνω σε απλές και σύνθετες ροές εργασίας πάνω σε γράφους μεγάλης κλίμακας. Παράλληλα, θα έχουμε την δυνατότητα να συγκρίνουμε ορισμένες από τις πιο δημοφιλείς πλατφόρμες επεξεργασίας γράφων και να εξάγουμε συμπεράσματα σχετικά με την επίδοσή τους.

1.1 Αντικείμενο της διπλωματικής

Σκοπός της διπλωματικής εργασίας είναι η μελέτη των δημοφιλέστερων συστημάτων επεξεργασίας γράφων σε μεγάλη κλίμακα καθώς επίσης και η σύνθεση συγκριτικής προτυποποίησης (Benchmarking) των συστημάτων για διαφορετικούς αλγορίθμους. Μέσω της προτυποποίησης θα δίνεται η δυνατότητα στον χρήστη να μπορεί να επιλέξει την κατάλληλη πλατφόρμα ανάλογα με την εφαρμογή που επιθυμεί να υλοποιήσει. Επιπλέον, η παρούσα διπλωματική στοχεύει να αποδείξει πειραματικά πως οι ροές εργασίας πάνω σε γράφους είναι εφικτό να έχουν ικανοποιητική επίδοση και κλιμακωσιμότητα πάνω σε πολυσυστηματικά περιβάλλοντα. Τέλος, η διπλωματική αποσκοπεί στη μελέτη και ύστερα στην ανάδειξη του IReS ως ένα σύστημα συντονισμού και εκτέλεσης πολύπλοκων εργασιών που υπερτερεί στην επίδοση έναντι των μεμονωμένων συστημάτων επεξεργασίας γράφων.

1.1.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Μελετήθηκαν τα συστήματα Apache Spark, Apache Flink, Apache Giraph, Neo4j και έγινε συγκριτική προτυποποίηση ανάμεσα στις τέσσερις πλατφόρμες σύμφωνα με την επίδοση αλλά και με την διαχείριση των πόρων της υποδομής.

2. Μελετήθηκε ο μετα-δρομολογητής IReS.

3. Αναλύθηκαν οι αλγόριθμοι γράφων: Page Rank, Weakly Connected Components, Single Source Shortest Paths.

4. Συγκρίθηκαν, μέσω του IReS, τα τέσσερα συστήματα ως προς το χρόνο εκτέλεσης, την χρήση του δικτύου καθώς επίσης και της χρήσης της Κεντρικής Μονάδας Επεξεργασίας και φαίνεται πως το Apache Spark πετυχαίνει κατά κόρον καλύτερες επιδόσεις.

5. Έγινε σύγκριση της επίδοσης των συστημάτων ως προς τέσσερις παραμέτρους: πλήθος ακμών, πυκνότητα γράφου, clustering coefficient και assortativity.

6. Μελετήθηκε η επίδοση των συστημάτων και ως βασικό συμπέρασμα προκύπτει η μη ύπαρξη ενός μόνο συστήματος που υπερτερεί έναντι των υπολοίπων για κάθε παράμετρο και αλγόριθμο.

7. Αποδεικνύεται πως το IReS με την πρόταση συνδυασμού των συστημάτων πετυχαίνει καλύτερες, και στη χειρότερη ίδιες επιδόσεις (ανεξάρτητα παραμέτρου), με το αν έτρεχε η ροή εργασίας σε ένα μόνο σύστημα. Η επίδοση φτάνει έως και το $x1.5$.

1.2 Οργάνωση του τόμου

Στην παρούσα υποενότητα παρουσιάζεται η δομή της διπλωματικής εργασίας καθώς επίσης και μία περιγραφή του τι περιλαμβάνει κάθε ενότητα από δω και στο εξής.

Συγκεκριμένα, το Κεφάλαιο 2 παρουσιάζει τις κατηγορίες στις οποίες διακρίνονται τα συστήματα επεξεργασίας γράφων καθώς και τα προγραμματιστικά μοντέλα που ακολουθεί καθένα από αυτά. Στο Κεφάλαιο 3 περιγράφονται εκτενώς τα συστήματα επεξεργασίας που πρόκειται να μελετήσουμε και να συγκρίνουμε. Εν συνεχεία, στο Κεφάλαιο 4 εξηγείται η έννοια του περιβάλλοντος πολλαπλών συστημάτων μελετώντας τον μετα-δρομολογητή IReS και την αρχιτεκτονική που τον διέπει. Επιπλέον, στο Κεφάλαιο 5 εξηγούνται τόσο οι αλγόριθμοι που θα υλοποιηθούν στα συστήματα όσο και η διαδικασία πειραματικής αξιολόγησης των συστημάτων. Τέλος, η διπλωματική εργασία καταλήγει στο Κεφάλαιο 6 όπου συνοψίζουμε τα αποτελέσματα που λάβαμε και συγκεντρώνουμε τις παρατηρήσεις που υπάρχουν πάνω στην πειραματική διαδικασία.

Κεφάλαιο 2

Συστήματα επεξεργασίας γράφων σε μεγάλη κλίμακα

2.1 Εισαγωγή

Τα συστήματα τα οποία έχουν αναπτυχθεί με σκοπό την κλιμακώσιμη επεξεργασία γράφων, παρουσιάζουν σημαντικές διαφορές μεταξύ τους. Αρχικά, η έρευνα επικεντρώθηκε σε κατανεμημένα περιβάλλοντα. Συγκεκριμένα, έγιναν αρκετές προσπάθειες επεξεργασίας γράφων πάνω σε περιβάλλον Hadoop [13] χρησιμοποιώντας το προγραμματιστικο πλαίσιο Map Reduce [3]. Ωστόσο, οι αλγόριθμοι που εφαρμόζονται πάνω σε γράφους είναι επαναληπτικοί και έχουν υψηλές απαιτήσεις σε αναδιανομή των δεδομένων και σε επικοινωνία. Δεδομένης της δυσκολίας του Map Reduce να ανταπεξέλθει στις παραπάνω δυσκολίες δημιουργήθηκε ένα σύνολο από συστήματα τα οποία υλοποιούν ποικίλα μοντέλα εκτέλεσης και δεδομένων. Με βάση τα μοντέλα που υλοποιούν, τα συστήματα επεξεργασίας γράφων διαχωρίζονται σε τρεις βασικές κατηγορίες:

i) Βάσεις δεδομένων για γράφους

ii) Κατανεμημένα συστήματα επεξεργασίας γράφων

iii) Κατανεμημένα συστήματα ροής δεδομένων σε μορφή γράφου

2.2 Βάσεις δεδομένων για γράφους

Η βάση δεδομένων για γράφους είναι μία βάση δεδομένων που δίνει ισόποση έμφαση τόσο στα δεδομένα όσο και στον τρόπο με τον οποίο αυτά σχετίζονται. Με άλλα λόγια, τα συστήματα που ανήκουν σε αυτήν την κατηγορία παρέχουν ένα πλούσιο μοντέλο αναπαράστασης δεδομένων καθιστώντας ευκολότερη την αποθήκευση και επεξεργασία των γράφων. Συγκεκριμένα, ο τρόπος με τον οποίο οι συγκεκριμένες βάσεις αντιμετωπίζουν τις σχέσεις καθιστά τη διάσχυση ενός γράφου πολύ πιο αποδοτική σε σχέση με τον τρόπο διαχείρισης από τις σχεσιακές βάσεις δεδομένων [10]. Μία γνωστή βάση για γράφους είναι η Neo4j [23].

2.2.1 Η ιδιότητα Γράφος

Κύριο χαρακτηριστικό των βάσεων δεδομένων για γράφους είναι ιδιότητα Γράφος (Property Graph Model). Πρόκειται για το μοντέλο με το οποίο τα δεδομένα αποθηκεύονται και επεξεργάζονται. Η *ιδιότητα Γράφος* αποτελείται από ένα σύνολο από κόμβους και ένα σύνολο από σχέσεις.

Οι κόμβοι αντιστοιχούν στις οντότητες της βάσης. Μπορούν να έχουν ένα πλήθος από ιδιότητες καθώς επίσης και από ετικέτες. Όσον αφορά τις σχέσεις, πρόκειται για σημασιολογικές συνδέσεις μεταξύ των κόμβων της βάσης. Κάθε σχέση έχει πεδία που καθορίζουν την πηγή, το είδος και τον προορισμό της σχέσης. Όπως και οι κόμβοι, έτσι και οι σχέσεις μπορούν να έχουν ιδιότητες.

2.2.2 Γλώσσα ερωτημάτων

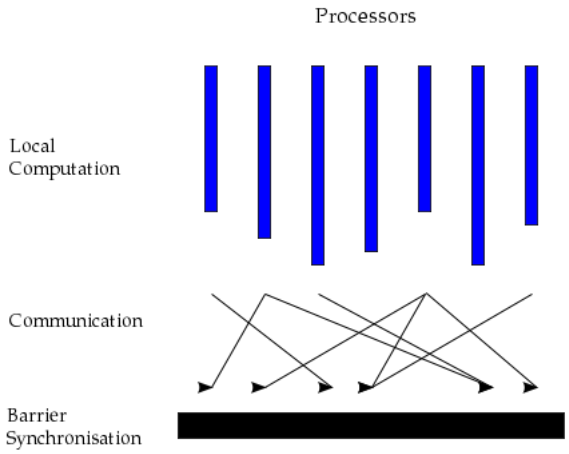
Η επεξεργασία των γράφων γίνεται με χρήση γλώσσας ερώτηματων αντίστοιχης της SQL στις σχεσιακές βάσεις δεδομένων. Κάθε βάση δεδομένων για γράφους παρέχει τη δικιά της γλώσσα ερωτημάτων. Τέσσερα βασικα ερωτήματα πάνω σε γράφους είναι τα εξής: γειτνίαση, προσβασιμότητα, αντιπαραβολή προτύπων και συγκέντρωση. Με αυτά τα ερωτήματα μπορούμε να βρούμε τους γείτονες ενός κόμβου, να ελέγξουμε την συνδετικότητα μεταξύ γράφων, να λάβουμε έναν υπογράφο της βάσης και να λάβουμε συγκεντρωτικές πληροφορίας για ιδιότητες του γράφου συνολικά. Όλες οι γλώσσες ερωτημάτων σε βάσεις γράφων υποστηρίζουν τους παραπάνω τελεστές. Η γλώσσα ερωτημάτων που χρησιμοποιεί το Neo4j καλείται Cypher.

2.3 Κατανεμημένα συστήματα επεξεργασίας γράφων

Όπως αναφέραμε και στην εισαγωγή, η ανάλυση γράφων αποτελείται κυρίως από επαναληπτικούς αλγορίθμους. Οι βάσεις δεδομένων, αν και παρουσιάζουν πολύ καλή επίδοση στην απάντηση ερωτημάτων, δε μπορούν να ανταπεξέλθουν εξίσου και στους επαναληπτικούς αλγορίθμους. Τα κατανεμημένα συστήματα επεξεργασίας γράφων στοχεύουν στη βέλτιστη υλοποίηση τέτοιων αλγορίθμων. Τα διάφορα προγραμματιστικά μοντέλα των παραπάνω συστημάτων βασίζονται στην αρχιτεκτονική αφέντη/εργάτη. Συγκεκριμένα, στο κατανεμημένο σύστημα ένας κόμβος έχει το ρόλο του αφέντη ως συντονιστή των εργασιών ενώ οι υπόλοιποι κόμβοι έχουν το ρόλο του εργάτη και εκτελούν τις αντίστοιχες εργασίες. Το βασικό μοντέλο επεξεργασίας ονομάζεται κομβο-κεντρικό μοντέλο. Σε αυτό το μοντέλο ο γράφος διαμοιράζεται σε όλους τους κόμβους-εργάτες. Κάθε εργάτης, για κάθε κόμβο του γράφου που λαμβάνει ως είσοδο αποθηκεύει τις ιδιότητες καθώς και τις ακμές από και προς τους γειτονικούς του κόμβους. Υπάρχουν αρκετές παραλλαγές αυτού του μοντέλου. Οι πιο σημαντικές είναι το μοντέλο BSP, το μοντέλο GAS καθώς και το μοντέλο SG.

2.3.1 Το μοντέλο εκτέλεσης BSP

Το μοντέλο εκτέλεσης Bulk Synchronous Parallel δημιουργήθηκε από τον Leslie Gabriel Valiant το 1990 [14]. Πρόκειται για ένα μοντέλο που έχει υλοποιηθεί απο πολλά προγραμμα-



Σχήμα 2.1: Το μοντέλο εκτέλεσης BSP

τιστικά πλαίσια και αντιμετωπίζει την υπολογιστική διαδικασία ως εργασίες που γίνοντα ανά κόμβο παράλληλα και ανεξάρτητα. Κάθε κόμβος μεταδίδει τα αποτελέσματά του στους υπόλοιπους μέσα από μία φάση μαζικής ανταλλαγής μηνυμάτων η οποία φράσσεται από ένα σημείο συγχρονισμού. Το σύνολο των εργασιών (υπολογισμοί και επικοινωνία) που γίνεται μεταξύ δύο σημείων συγχονισμού καλείται υπερβήμα. Στο Σχήμα 5.3 βλέπουμε τη ροή εκτέλεσης ενός υπερβήματος. Το πιο γνωστό σύστημα που υλοποιεί το μοντέλο BSP είναι το Pregel και στην οικογένεια αυτή περιλαμβάνεται και το σύστημα Apache Giraph που πρόκειται να μελετήσουμε.

2.3.2 Το μοντέλο εκτέλεσης GAS

Το μοντέλο Gather-Apply-Scatter πρόκεται για ένα κόμβο-κεντρικό μοντέλο που αποτελείται από τρεις φάσεις. Αρχικά, ορίζεται η φάση της Συγκέντρωσης όπου συγκεντρώνονται τα μηνύματα που που προορίζονται για κάθε κόμβο ενός εργάτη. Κατά τη φάση της Εφαρμογής, βάσει του αποτελέσματος της συγκέντρωσης ενημερώνεται η τιμή του κατάλληλου κόμβου. Τέλος, στη φάση του Διασκορπισμού, δημιουργούνται τα μηνύματα που πρόκειται να στείλει κάθε κόμβος και περιέχουν την ενημερωμένη τιμή του. Σε αυτό το μοντέλο, η παραλληλοποίηση γίνεται σε επίπεδο ακμών.

Σημαντικό κριτήριο για την επίδοση ενός κομβο-κεντρικού μοντέλου επεξεργασίας είναι ο τρόπος τμηματοποίηση του γράφου που λαμβάνεται ως είσοδος και ο διαμοιρασμός του στου εργάτες του κατανεμημένου συστήματος. Το μοντέλο GAS είναι αρκετά αποτελεσματικό όταν οι γράφοι έχουν μεγάλη απόκλιση στην κατανομή των ακμών τους στους κόμβους. Αυτό οφείλεται στο γεγονός ότι, σε αντίθεση με το μοντέλο BSP, στο GAS η υπολογιστική διαδικασία ενός κόμβου διαμοιράζεται στους εργάτες του συστήματος. Ωστόσο, το μοντέλο GAS απαγορεύει την άμεση επικοινωνία μη γειτονικών κόμβων με αποτέλεσμα να περιορίζει την εκφραστικότητα σε πιο γενικά μοτίβα επικοκινωνίας.

2.3.3 Το μοντέλο εκτέλεσης SG

Το Scatter-Gather (ή αλλιώς Signal-Collect [19]) αποτελεί άλλο ένα κόμβο-κεντρικό προγραμματιστικό μοντέλο. Όπως και το BSP, το SG λειτουργεί πάνω σε μεγάλα σύγχρονα βήματα. Κεντρική διαφορά μεταξύ των δύο αποτελεί ο διαχωρισμός της συνάρτησης κόμβου σε δυο φάσεις: τη φάση Διασκορπισμού (Scatter) και τη φάση Συγκέντρωσης (Gather).

Κατά τη φάση του Διασκορπισμού, κάθε κόμβος-εργάτης της συστάδας φορτώνει στη μνήμη το σύνολο των κόμβων που έχουν διαμοιραστεί ενώ παράλληλα οι ακμές αποθηκεύονται σε μεγάλους buffers με τη μορφή ροής δεδομένων. Με αυτόν τον τρόπο, μέσω των ακμών, όσες ενημερώσεις γίνονται στην τιμή των κόμβων προωθούνται και στους γείτονικούς τους κόμβους. Στη φάση της Συγκέντρωσης, αφού φορτωθεί το σύνολο των κόμβων στη μνήμη αποθηκεύεται σε buffers το σύνολο ενημερώσεων που προέκυψαν στο προηγούμενο στάδιο σε μορφή ροής δεδομένων. Καθώς επεξεργάζονται οι ενημερώσεις, οι τιμές συγκεντρώνονται (μέσω τελεστή) στους κόμβους-παραλλήπτες. Και στις δύο φάσεις χρησιμοποιούνται πολλές δομές προσωρινής μνήμης (buffers) προκειμένου να γίνονται επικαλύψη μεταξύ υπολογισμών και επικοινωνίας.

Χαρακτηριστικό του συγκεκριμένου μοντέλου αποτελεί το γεγονός πως η αποστολή και λήψη των μηνυμάτων ενός κόμβου συμβαίνει στο ίδιο υπερβήμα. Με άλλα λόγια, κατά τη διάρκεια του υπερβήματος i, η συνάρτηση Gather του κόμβου v έχει πρόσβαση στα μηνύματα που στάλθηκαν κατά τη συνάρτηση Scatter στην επανάληψη i.

2.4 Κατανεμημένα συστήματα ροής δεδομένων σε μορφή γράφου

Μία επιπλέον προσέγγιση σχετικά με την επεξεργασία γράφων αφορούσε το ζήτημα της αποθήκευσης και τροποποίησης των δεδομένων κατά τη διάρκεια μίας ροής εργασιών. Η ανάγκη για συνδυασμό επεξεργασίας τόσο αδόμητων δεδομένων όσο και γράφων σε ένα προγραμματιστικό πλαίσιο οδήγησε στη δημιουργία των κατανεμημένων συστημάτων ροής δεδομένων. Η υψηλή επίδοση των παραπάνω συστημάτων έγκειται στον αποδοτικό τρόπο με τον οποίο αποθηκεύουν τα δεδομένα κατανεμημένα στη μνήμη. Συστήματα όπως το Apache Spark [20] και το Apache Flink [21], παρέχουν στο χρήστη τη δυνατότητα να χειριστούν το γράφο είτε ως αδόμητο σύνολο δεδομένων με τη βοήθεια τελεστών γενικού σκοπού, είτε ως σύνολο από κόμβους και ακμές.

2.4.1 Μοντέλοποίηση γράφου

Σε αυτού του είδος τα συστήματα ο γράφος G ορίζεται ως $G = (V, E)$ με V το σύνολο των κόμβων και E το σύνολο των ακμών. Το Σχήμα 2.2 παρουσιάζει τον τρόπο με τον οποίο ορίζεται η κλάση γράφος στο σύστημα Apache Flink. Παρατηρούμε ότι η κλάση Graph αποτελείται από ένα σύνολο κόμβων vertices το οποίο έχει κάποιον αριθμό-ταυτότητα του συστήματος με τύπο K καθώς επίσης και ένα πεδίο που αντιστοιχεί στην τιμή που έχει ο

κάθε κόμβος του γράφου. Αντίστοιχα, ορίζεται και το σύνολο ακμών edges όπου υπάρχουν οι πληροφορίες της κατεύθυνσης αλλά και της τιμής κάθε ακμής.

```
class Graph<K, VV, EV> {
    DataSet<Vertex<K, VV>> vertices
    DataSet<Edge<K, EV>> edges
}
```

Σχήμα 2.2: Η κλάση Graph στο προγραμματιστικό πλαίσιο Gelly του συστήματος Apache Flink.

2.4.2 Το επαναληπτικο μοντέλο εκτέλεσης

Το επαναληπτικό μοντέλο εκτέλεσης συνίσταται στην εφαρμογή μίας "βηματικής" συνάρτησης πάνω σε ένα σύνολο δεδομένων και η ενσωμάτωσή της σε έναν επαναληπτικό τελεστή. Θα μελετήσουμε το παράδειγμα του συστήματος Apache Flink αλλά κατα αντιστοιχία ορίζονται παρόμοιες συναρτήσεις και στα υπόλοιπα συστήματα. Οι επαναληπτικοί τελεστές διακρίνονται σε απλούς και δ-επαναληπτικούς τελεστές. Βασική διαφορά είναι το γεγονός ότι στους απλούς τελεστές ως είσοδο έχουμε το γράφο και σε κάθε επανάληψη εφαρμόζουμε τη "βηματική" συνάρτηση σε ολόκληρο το σύνολο δεδομένων της προηγούμενης επανάληψης. Αντίθετα, στους δ-επαναληπτικούς τελεστές ως είσοδος λαμβάνεται το σύνολο δεδομένων καθώς επίσης και ένα σύνολο λύσης το οποίο εξελίσσεται κατά τη διάρκεια των επαναλήψεων. Κατά την εφαρμογή της "βηματικής" συνάρτησης κρατείται μόνο το σύνολο δεδομένων που μεταβάλλεται στην παρούσα επανάληψη και μόνο αυτό το σύνολο τροφοδοτείται στην επόμενη. Αυτός ο τελεστής καλύπτει την περίπτωση των αυξητικών επαναλήψεων όπου αντί να επανυπολογίζεται όλο το σύνολο δεδομένων, κρατάμε μόνο το υποσύνολο που μας ενδιαφέρει. Με αυτόν τον τρόπο οδηγούμαστε σε ποιο αποδοτικούς αλγορίθμους. Στο Σχήμα 2.3 βλέπουμε τη διαφορά της ροής των δύο τελεστών που περιγράψαμε.

Σχήμα 2.3: Σύγκριση επαναληπτικών μοντέλων

Κεφάλαιο 3

Τεχνική ανάλυση συστημάτων

3.1 Εισαγωγή

Σε αυτήν την ενότητα θα παρουσιάσουμε λεπτομερώς τα συστήματα με τα οποία θα ασχοληθούμε. Όπως είδαμε και στην Ενότητα 2, υπάρχουν τρεις κατηγορίες συστημάτων επεξεργασίας γράφων. Λαμβάνοντας υπόψην τον παραπάνω διαχωρισμό, διαλέξαμε τα συστήματα ανοιχτού κώδικα έτσι ώστε να αντιπροσωπεύουν κάθε κατηγορία. Τα συστήματα αυτά είναι:

- Apache Spark
- Apache Flink
- Apache Giraph
- Neo4j

Στον Πίνακα 3.1 βλέπουμε την κατάταξη των συστημάτων ανά κατηγορία. Διαλέξαμε δύο συστήματα από την κατηγορία των συστημάτων ροής δεδομένων καθώς το Spark και το Flink είναι ευρέως διαδεδομένα.

Κατηγορίες συστημάτων για γράφους	**Spark**	**Flink**	**Giraph**	**Neo4j**
Βάση δεδομένων				✓
Επεξεργασίας γράφων			✓	
Ροής δεδομένων	✓	✓		

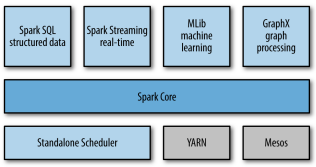
Πίνακας 3.1: Κατηγοριοποίηση των προς εξέταση συστημάτων

3.2 Apache Spark

Το Apache Spark [20] είναι μία ενιαία μηχανή επεξεργασίας κατανεμημένων δεδομένων που δημιουργήθηκε στο AMPLab του University of California, Berkeley το 2009. Το Spark

καλύπτει ένα μεγάλο εύρος ειδών επεξεργασίας δεδομένων που παλαιότερα χρειάζονταν ξεχωριστές μηχανές (εξού και ο όρος ¨ενιαία¨). Το εύρος αυτό περιλαμβάνει επεξεργασία γράφων, ροής δεδομένων, μηχανικής εκμάθησης καθώς και επεξεργασία με SQL ερωτήματα.

Το προγραμματιστικό μοντέλο του Spark επεκτείνει το μοντέλο του Map Reduce με την έννοια των Resilient Distributed Dataset (RDD). Πρόκειται για συλλογές από αντικείμενα τα οποία βρίσκονται καταμερισμένα στη συστάδα και μπορούν να επεξεργαστούν παράλληλα. Τα RDDs προκύπτουν εάν εφαρμόσουμε στα δεδομένα μας μία λειτουργία που ονομάζεται μετασχηματισμός. Το Spark χρησιμοποιεί τη στρατηγική της οκνηρής αποτίμησης πάνω στα RDDs γλυτώνοντας άσκοπους υπολογισμούς, αποτιμώντας ολόκληρο το γράφο μετασχηματισμών τη στιγμή της εκτέλεσης. Βασικά χαρακτηριστικά του RDD είναι η δυνατότητα διαμοιρασμού των δεδομένων καθώς και η ανοχή σε σφάλματα. Το RDD παρέχει υποστήριξη για διαμοιρασμό δεδομένων στους κόμβους της συστάδας μεταξύ των υπολογισμών. Αυτή είναι και η βασική διαφορά στην έναντι του Map Reduce στο οποίο ανάμεσα στους υπολογισμούς τα αποτελέσματα πρέπει να γράφονται στο δίσκο. Όσον αφορά στην ανοχή παραδοσιακά χρησιμοποιούταν η τεχνική των αντιγράφων των δεδομένων σε περισσότερους από έναν κόμβο. Ωστόσο, στο Spark για κάθε RDD αποθηκεύεται ο γράφος των μετασχηματισμών από τον οποίο αυτό προήλθε. Επομένως, αντί να μεταφέρουμε δεδομένα στο δίκτυο μέσω των αντιγράφων, σε περίπτωση απώλιας τμήματος του RDD λόγω αποτυχίας ενός κόμβου, ξανατρέχουμε το γράφο καταγωγής του RDD από την αρχή.



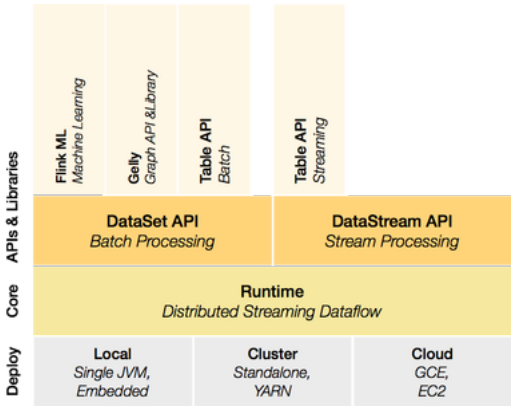
Σχήμα 3.1: Η αρχιτεκτονική του συστήματος Apache Spark

Αντίστοιχα με το Map Reduce, ορίζεται και η Spark συστάδα πάνω από ένα κατανεμημένο σύστημα αρχείων Hadoop, στην οποία υπάρχει ένας κόμβος (Driver) που είναι υπεύθυνος για τη δρομολόγηση και ανάθεση των εργασιών και οι κόμβοι-εργάτες (Workers) που είναι υπεύθυνοι για τον υπολογισμό των εργασιών και την αποθήκευση των δεδομένων.

Στο Spark έχει αναπτυχθεί ένα σύνολο από βιβλιοθήκες υψηλού επιπέδου που στοχεύουν στην κάλυψη των περισσότερων σεναρίων που καλύπτουν οι εξειδικευμένες υπολογιστηκές μηχανές. Η βιβλιοθήκη που θα χρησιμοποιήσουμε στην παρούσα διπλωματική εργασία είναι η GraphX. Η βιβλιοθήκη GraphX [24] παρέχει διεπιφάνεια παρόμοια με των εξειδικευμένης μηχανών Pregel και GraphLab [17] εφαρμόζοντας τις ίδιες βελτιστοποιήσεις που έχουν και εκείνες (πχ. διαμερισμός κόμβου). Η GraphX μας παρέχει το μοντέλο δεδομένων Graph το οποίο αποτελείται απο συλλογές ζευγαριών κλειδί-τιμή και όπως βλέπουμε και στο σχήμα αποτελούν το σύνολο των κόμβων και το σύνολο των ακμών του γράφου $G = (V, E)$. Με αυτόν τον τρόπο, μπορούμε να αξιοποιήσουμε την παραλληλοποίηση που μας προσφέρει τόσο το Spark με τους μετασχηματισμούς πάνω σε συλλογές δεδομένων όσο και οι εξειδικευμένες μηχανές επεξεργασίας γράφων με τις ιδιότητες που προσδίδουν στους γράφους.

3.3 Apache Flink

Το Apache Flink [21] πρόκειται για ένα σύστημα που συνδυάζει την επεξεργασίας ροής δεδομένων με το batch processing και κατασκευάστηκε από την Apache Software Foundation.



Σχήμα 3.2: Η αρχιτεκτονική του συστήματος Apache Flink

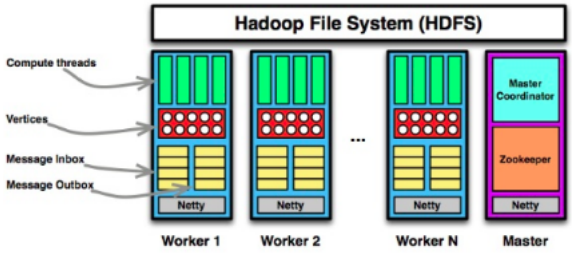
Ο πυρήνας του συστήματος Apache Flink είναι μηχανή κατανεμημένης ροής δεδομένων που εκτελεί προγράμματα ροής δεδομένων. Ένα τέτοιο πρόγραμμα Flink αποτελείται από έναν Ακυκλικό Κατευθυνόμενο Γράφο όπου αποτελείται από τελεστές που συνδέονται μεταξύ τους με ροές δεδομένων. Δεδομένης της παραλληλοποιήσιμότητας των τελεστών, αυτοί μοιράζονται στους κόμβους της συστάδας σε υποεργασίες με τις ροές δεδομένων να διαμοιράζονται σε καθεμία απο αυτές. Προκειμένου να επικοινωνήσει ο πυρήνας του Flink με το κάθε είδος επεξεργασίας, παρέχονται οι κατάλληλες διεπιφάνειες. Για επεξεργασία ροής δεδομένων παρέχεται η διεπιφάνεια DataStream ενώ για την batch επεξεργασία η διεπιφάνεια DataSet στην οποία και θα επικεντρωθούμε. Στη διεπιφάνεια DataSet, υπάρχουν όλα τα απαραίτητα εργαλεία προκειμένου να γίνουν μετασχηματισμοί των δεδομένων καθώς επίσης ορίζονται πηγές και καταβόθρες που προσδιορίζουν το από που θα διαβαστούν τα δεδομένα και που θα καταλλήξουν (αποθήκευση σε αρχείο ή στην κονσόλα). Ένα Dataset αντικείμενο δεν υπάρχει υλοποιημένο στη μνήμη, όπως το RDD στο Spark, αλλά αντιπροσωπεύει μία ροή δεδομένων από μία πηγή ή τελεστή σε έναν τελεστή ή καταβόθρα. Για το Flink, το batch processing αντιμετωπίζεται ως υποπερίπτωση της επεξεργασίας ροής δεδομενων θεωρόντας τη ροή δεδομένων ως πεπερασμένη.

Όσον αφορά την ανοχή σε σφάλματα το Apache Flink υλοποιεί σημεία ελέγχου μέσω λήψης στιγμιότυπων των παράλληλων τελεστών που εκτελούνται. Ο μηχανισμός του Flink αυτός καλείται Asynchronous Barrier Snapshotting και πρόκειται για παραλλαγή του αλγορίθμου Chandy-Lamport [25].

Η δομή μίας αυτόνομης Flink συστάδας πάνω σε Hadoop είναι παρόμοια με αυτή του Spark. Το σύστημα ακολουθεί master/slave αρχιτεκτονική με τον έναν κόμβο (JobManager) να είναι υπεύθυνος για τη δρομολόγηση εργασιών, συντονισμό των σημείων ελέγχου και συντονισμό σε περίπτωση σφάλματος στη συστάδα και των υπόλοιπων κόμβων - εργατών (TaskManager) οι οποίοι φορτώνουν τις ροές δεδομένων και τις επεξεργάζονται.

Όπως βλέπουμε και στο Σχήμα 3.2 , στο Flink έχουν αναπτυχθεί ορισμένες βιβλιοθήκες υψηλού επιπέδου. Στην παρούσα διπλωματική θα ασχοληθούμε με την Gelly. Η Gelly περιλαμβάνει ένα σύνολο από μεθόδους οι οποίες διευκολύνουν την επεξεργασία δεδομένων σε μορφή γράφου. Σε αυτήν τη βιβλιοθήκη ένας γράφος αποτελείται από δύο DataSet αντικείμενα: τους κόμβους και τις ακμές. Συγκεντρωτικά, λοιπόν, η βιβλιοθήκη Gelly περιλαμβάνει μεθόδους για μετασχηματισμούς πάνω σε γράφους, μεθόδους για επαναληπτική επεξεργασία (βλ. Ενότητα 2) καθώς επίσης και ένα σύνολο από υλοποιημένους τελεστές πάνω σε γράφους. Χάρη στο Gelly μπορούμε σε μία πλατφόρμα να εκτελέσουμε μία αλληλουχία από εργασίας που αφορούν την προεργασίας στα δεδομένα, τη δημιουργία του γράφου και την αναλυτική πάνω σε αυτόν.

3.4 Apache Giraph



Σχήμα 3.3: Η αρχιτεκτονική του συστήματος Apache Giraph

Το Apache Giraph [22] είναι ένα έργο ανοιχτού κώδικα που μιμείται τη λειτουργία του συστήματος Pregel [26]. Πρόκειται για ένα κλιμακώσιμο σύστημα που παρουσιάστηκε το 2010 από τη Google ως πλατφόρμα υλοποίησης παράλληλων αλγορίθμων. Οι αλγόριθμοι που υλοποιούνται στο Apache Giraph βασίζονται στην έννοια της κόμβο-κεντρικής προσέγγισης που προήλθε από το μοντέλο BSP.

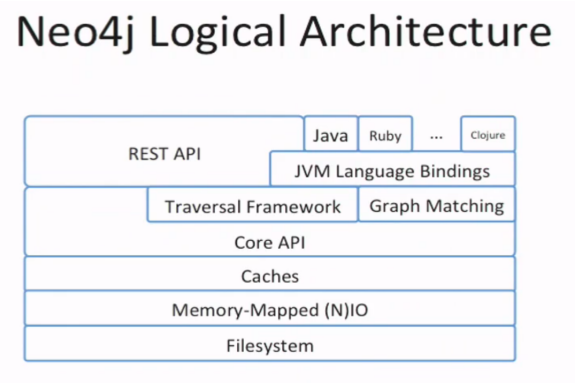
Η επεξεργασία του γράφου γίνεται μέσα από μία ακολουθία επαναλήψεων που τα αποκαλούμε υπερβήματα - supersteps. Κατά τη διάρκεια ενός βήματος S εκτελείται παράλληλα μία συνάρτηση για κάθε κόμβο $v \in V$. Αυτή η συνάρτηση ορίζεται από το χρήστη και σκοπός της είναι να διαβάσει τα μηνύματα που στάλθηκαν στο κόμβο v στο βήμα $S-1$, να στείλει μηνύματα από τον κόμβο προς άλλους κόμβους οι οποίοι θα τα διαβάσουν στο βήμα $S+1$ και να μεταβάλλει την τιμή/κατάσταση του κόμβου v και των εξερχόμενων ακμών του. Το κάθε βήμα λογίζεται ως ατομική μονάδα του παράλληλου υπολογισμού. Αν ένας κόμβος στο γράφο δεν έχει να στείλει κάποιο μήνυμα κατά το υπερβήμα τότε θέτει την κατάστασή του ως ανενεργή και επανενεργοποιείται μόνον αν σε επόμενο υπερβήμα λάβει κάποιο εισερχόμενο μήνυμα. Μία εφαρμογή τερματίζει είτε αν όλοι ο κόμβοι του γράφου έχουν τεθεί σε ανενεργή κατάσταση, είτε όταν συμπληρώσουμε ένα πλήθος υπερβημάτων που έχουμε ορίσει εκ των προτέρων ως άνω όριο.

Το Apache Giraph λειτουργεί πάνω σε Hadoop συστάδα και εκτελείται ως Hadoop ερ-

γασία. Ανάλογα με την έκδοση του Hadoop το Giraph μπορεί να λειτουργήσει είτε ως μία εφαρμογή Map Reduce μόνο από μεθόδους map (map-only) είτε μεσω του YARN [27] και τη δημιουργία containers. Όπως βλέπουμε στο Σχήμα 3.3 το Apache Giraph ακολουθεί αρχιτεκτονική master-slave. Ιδιαίτερα, σε κάθε κόμβο εργάτη αποθηκεύεται ένα υποσύνολο των κόμβων του γράφου καθώς επίσης και τα εισερχόμενα και εξερχόμενα μηνύματα που αυτοί οι κόμβοι λαμβάνουν ανά υπερβήμα. Η αρμοδιότητα του κόμβου αφέντη είναι να προωθεί την υπολογιστική διαδικασία από το ένα υπερβήμα στο άλλο, να παρακολουθεί την υγεία των κόμβων εργατών, να διαμοιράζει τον γράφο εισόδου στους ενεργούς κόμβους εργάτες καθώς και να τρέχει εργασίες που ορίζονται από το χρήστη (πχ. συγκέντρωση των μηνυμάτων). Επιπλέον, μία λειτουργία που συνήθως αναλαμβάνει ο κόμβος αφέντης είναι ο συντονισμός του συνόλου των κόμβων κατά την υπολογιστική διαδικασία. Σε περιβάλλον Hadoop η υπηρεσία του συντονισμού επιτυγχάνεται μέσω του Apache Zookeeper [28].

3.5 Neo4j

Το Neo4j [23] είναι μία ανοιχτού κώδικα, NoSQL φυσική βάση δεδομένων για γράφους που δημοσιεύτηκε το 2007. Ονομάζεται έτσι καθώς μπορεί αποδοτικά να αναπαραστήσει την ιδιότητα γράφου μέχρι και το επίπεδο αποθήκευσης. Αυτό σημαίνει ότι ο γράφος αποθηκεύεται στη βάση ακριβώς όπως ορίστηκε από τον χρήστη και η διάσχισή του επιτυγχάνεται με χρήση δεικτών. Συνεπώς, ερωτήματα τα οποία ήταν ¨ακριβά' για μία σχεσιακή βάση μπορούν να γίνουν αποδοτικά με το Neo4j. Τέτοιου είδους ερωτήματα αποτελεί η διάσχιση ενός γράφου. Επιπλέον, το Neo4j χαρακτηρίζεται από την ευελιξία του στην αναπαράσταση δεδομένων. Οποιοδήποτε σύνολο δεδομένων χαρακτηρίζεται από συσχετίσεις μεταξύ των στοιχείων του μπορεί να αναπαρασταθεί στην πλατφόρμα Neo4j. Παρέχεται, επίσης, και η ευελιξία στην αναπαράσταση των δεδομένων χάρη στην ιδιότητα Γράφου που περιγράψαμε στην Ενότητα 2. Επιπλέον, μία βάση Neo4j παρουσιάζει και χαρακτηριστικά τυπικών βάσεων. Συγκεκριμένα, μία βάση Neo4j προσφέρει ACID συναλλαγές, πολιτική αντιμετώπισης σφαλμάτων κατά την εκτέλεση κ.α.



Σχήμα 3.4: Η αρχιτεκτονική του συστήματος Apache Neo4j

Η επίδοση του Neo4j συνίσταται σε πολλούς παράγοντες πέρα από την παραλληλοποιη-

σιμότητα του εκάστοτε αλγόριθμου. Το σύστημα παρέχει σύνολα εντολών υψηλού επιπέδου (API) τα οποία ενισχύουν την επίδοση σε διάφορα στάδια της επεξεργασίας ενός γράφου (Σχήμα 3.4). Αρχικά, υπάρχει ένα σύνολο εντολών το οποίο είναι υπεύθυνο να φορτώνεται παράλληλα ο γράφος στη βάση και αποδοτικά. Έπειτα, το Traversal API παρέχει στον χρήστη τη δυνατότητα να ορίζει τον τρόπο διάσχισης στο γράφο λαμβάνοντας βέλτιστα μόνον εκείνες οι πληροφορίες του γράφου (τιμή κόμβων, ακμών) που χρειάζεται ο αλγόριθμος. Συναντάμε και το Core (Graph) API το οποίο μεταξύ άλλων υλοποιεί και την ιδιότητα γράφου που περιγράψαμε στην Ενότητα 2. Ένα από τα χαρακτηριστικά που προσφέρει το Neo4j είναι η γλώσσα προγραμματισμού Cypher. Πρόκειται για μία δηλωτική γλώσσα προγραμματισμού παρόμοια με την SQL αλλά βελιστοποιημένη στα ερωτήματα γράφων. Το Neo4j έχει τη βιβλιοθήκη Neo4j Graph Algorithms η οποία προσφέρει αποδοτικούς παράλληλους αλγορίθμους επεξεργασίας γράφων οι οποίοι έχουν εκφραστεί σε διαδικασίες της γλώσσας Cypher. Όσον αφορά την επίδοσή, η πλατφόρμα παρουσιάζει εκπληκτικά αποτελέσματα (σε αντίθεση με τις σχεσιακές βάσεις) όσο το βάθος και η πολυπλοκότητα των σχέσεων αυξάνεται. Τέλος, το Neo4j δίνει τη δυνατότητα στο χρήστη να δει οπτικοποιημένα ολόκληρο τον γράφο χαρακτηριστικό που παρέχουν ελάχιστες πλατφόρμες επεξεργασίας.

Το Neo4j τρέχει κεντρικά σε έναν υπολογιστή, ωστόσο σε συγκεκριμένες εκδόσεις υποστηρίζεται και λειτουργία σε συστάδα με σκοπό την υψηλή διαθεσιμότητα σε πραγματικά σενάρια χρήσης.

Κεφάλαιο 4

Πολυ-συστηματική επεξεργασία

4.1 Εισαγωγή

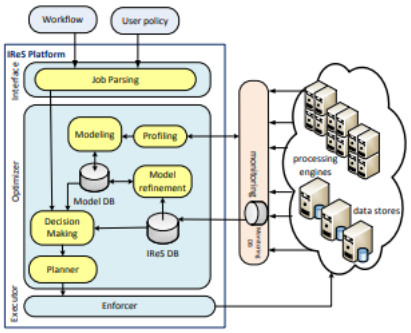
Στα προηγούμενα κεφάλαια μελετήσαμε την πληθώρα των συστημάτων επεξεργασίας γράφων που υπάρχουν. Έρευνες [6], [7], [8], [9] δείχνουν πως δεν υπάρχει ένα σύστημα επεξεργασίας γράφων το οποίο να υπερτερεί έναντι των υπολοίπων. Στο κεφάλαιο 3, είδαμε πως το Giraph απαιτεί συγκεκριμένη μορφή δεδομένων προκειμένου να αποδώσει στους επαναληπτικούς αλγορίθμους. Είδαμε επίσης ότι το Neo4j ως βάση δεδομένων είναι βελτιστοποιημένη για ερωτήματα που αφορούν τη διάσχιση σε γράφο ενώ δεν αποδίδει εξίσου σε επαναληπτικούς αλγορίθμους. Επιπλέον, όσον αφορά τα μοντέλα εκτέλεσης, παρατηρήσαμε ότι το GAS μοντέλο υπερτερεί έναντι του BSP μοντέλου όταν ο γράφος που μελετάμε παρουσιάζει μεγάλη απόκλιση στην κατανομή των ακμών στους κόμβους. Βλέπουμε, λοιπόν, πως δεν υπάρχει ένα μοντέλο αναπαράστασης δεδομένων ή ένα μοντέλο εκτέλεσης το οποίο να μπορεί να καλύψει αποτελεσματικά όλες τις ανάγκες ανάλυσης γράφων.

Λαμβάνοντας υπόψιν τα παραπάνω και σε συνδυασμό με το γεγονός ότι οι σύγχρονες απαιτήσεις της επεξεργασίας γράφων είναι υψηλές, προτάθηκε η έννοια της συνύπαρξης ενός συνόλου από συστήματα επεξεργασίας σε ένα κοινό περιβάλλον [11]. Επιπλέον, προκειμένου να είναι εφικτή και αποδοτική η αξιοποίηση των συστημάτων επεξεργασίας, προτάθηκε η σύσταση ενός πλαισίου [12] το οποίο θα λειτουργεί ως διαχειριστής των συστημάτων και συντοντιστής των ροών εργασίας βάσει ορισμένων κριτηρίων που ορίζονται από τον χρήστη. Ένα σύστημα που υλοποιεί το ζητούμενο πλαίσιο είναι το Intelligent Resource Scheduler (IReS).

4.2 Ο μετα-δρομολογητής IReS

Το IReS [29] αποτελεί ένα ολοκληρωμένο σύστημα ανοιχτού κώδικα το οποίο είναι υπεύθυνο για την διαχείριση, εκτέλεση και παρακολούθηση πολύπλοκων ροών εργασίας πάνω σε περιβάλλον πολλών συστημάτων επεξεργασίας. Συγκεκριμένα, η λειτουργία του IReS συνίσταται στην εύρεση και εκτέλεση της βέλτιστης δυνατής ροής εργασίας μελετώντας ένα σύνολο από τελεστές, δεδομένα καθώς επίσης και τις εξαρτήσεις που έχουν μεταξύ τους. Η βασική έννοια πίσω από το IReS είναι η μοντελοποίηση των διαθέσιμων τελεστών κάθε πλατ-

φόρμας βάσει του κόστους και χαρακτηριστικών επίδοσης. Ο χρήστης καλείται να ορίσει την πολιτική με την οποία το IReS θα προτείνει και θα εκτελέσει μία ροή εργασίας αναθέτοντας κάθε υποεργασίας στο πλέον επωφελές σύστημα επεξεργασίας. Ο λόγος για τον οποίο το IReS μπορεί να καλύψει πληθώρα συστημάτων επεξεργασίας και εργασίων είναι γιατί αντιμετωπίζει τους τελεστές ως μαύρο κουτί. Συγκεκριμένα, αρκεί να ορίσουμε μία περιγραφή καθώς επίσης και ένα μοντέλο κόστους της κάθε εργασίας πάνω στα διαθέσιμα συστήματα επεξεργασίας. Στο Σχήμα 4.1 βλέπουμε την αρχιτεκτονική του συστήματος η οποία αποτελείται από τρία επίπεδα: το επίπεδο διεπιφάνειας, το επίπεδο βελτιστοποίησης και το επίπεδο εκτέλεσης.



Σχήμα 4.1: Η αρχιτεκτονικη του συστήματος IReS

Η διεπιφάνεια είναι υπεύθυνη για την επικοινωνία μεταξύ χρήστη και συστήματος. Ο χρήστης σε αυτό το επίπεδο καλείται να ορίσει αναλυτικά τα δεδομένα, τους τελεστές και τις ροές εργασίας που θέλει να εκτελέσει προσδιορίζοντας τυχόν περιορισμούς και εξαρτήσεις. Επιπλέον έχει τη δυνατότητα να επιλέξει μεταξύ δύο ειδών περιγραφής δεδομένων και τελεστών: υλοποιημένη και αφαιρετική. Στην υλοποιημένη περιγραφή ο χρήστης καλείται να προσδιορίζει επακριβώς τον τρόπο με τον οποίο εκτελελείται ο τελεστής ή το μέρος στο οποίο βρίσκεται το αρχείο δεδομένων. Η αφαιρετική περιγραφή χρησιμοποιείται όταν θέλουμε να ορίσουμε μία ροή εργασίας και στη συνέχεια να τη δώσουμε στο IReS για να βρει τη βέλτιστη υλοποιημένη περιγραφή βάσει των κριτηρίων που θα του δώσουμε.

Το επίπεδο βελτιστοποίησης είνα υπεύθυνο για την επιλογή της βέλτιστης ροής εργασίας δεδομένης της πολιτικής που έχει οριστεί από τον χρήστη. Η διαδικασία επιλογής γίνεται σε πραγματικό χρόνο και για κάθε υποεργασία αφορά την επιλογή του κατάλληλου συστήματος που θα την εκτελέσει, το πλήθος των πόρων που θα διατεθούν, τον έλεγχο μετακίνησης δεδομένων από ένα σύστημα σε ένα άλλο καθώς επίσης και την τοποθεσία που θα αποθηκευτεί η έξοδος. Αυτό επιτυγχάνεται κατόπιν σύγκρισης των μοντέλων όλων των πιθανών διαθέσιμων συστημάτων για κάθε υποεργασία. Τα μοντέλα αυτά προκύπτουν από τη διαδικασία χαρακτηρισμού (profiling) των τελεστών κάθε συστήματος η οποία γίνεται εκ των προτέρων από το χρήστη με τη βοήθεια του συστήματος παρακολούθησης που παρέχει πληροφορίες για την επίδοση των τελεστών. Συγκεκριμένα, ορίζουμε τα χαρακτηριστικά (features) που θέλουμε να έχει η διαδικασία χαρακτηρισμού. Για παράδειγμα, μπορούμε να επιλέξουμε ως χαρακτηριστικό το πλήθος των ακμών που έχει ο γράφος και ως μεταβλητή στόχου (goal variable) τον χρόνο εκτέλεσης. Στη συνέχεια, αφού συγκεντρωθούν αρκετά σημεία για διαφορετικές τιμές

των χαρακτηριστικών, το IReS εκπαιδεύει μία σειρά από μοντέλα μηχανικής εκμάθησης και κρατάει αυτό με το μικρότερο σφάλμα (συνάρτηση λάθους).

Αφού βρεθεί η βέλτιστη ροή εργασίας, το IReS την εκτελεί μέσω του επιπέδου εκτέλεσης. Σε αυτό το σημείο, το σύστημα εφαρμόζει το βέλτιστο πλάνο εργασίας πάνω στο σύνολο των κόμβων της συστάδας. Συγκεκριμένα μεταφράζει κάθε βήμα του πλάνου σε API κλήσεις που αφορούν το εκάστοτε σύστημα επεξεργασίας.

Το σύστημα IReS έχει υλοποιηθεί σε Java και στηρίζεται στο YARN πλαίσιο [27] το οποίο είναι υπεύθυνο για τη διαχείριση, δέσμευση και δρομολόγηση των πόρων μίας συστάδας υπολογιστών αναθέτοντας σε κάθε εργασία ένα τμήμα των πόρων το οποίο καλείται container. Επιπλέον το IReS, προκειμένου να συντονίσει ολόκληρη τη ροή εργασίας, χρησιμοποιεί και επεκτείνει το Cloudera Kitten [30]. Πρόκειται για ένα σύνολο εργαλείων που βοηθούν στην ρύθμιση και εκτέλεση των containers και γενικό στην απλούστερη υλοποίηση YARN εφαρμογών.

Κεφάλαιο 5

Πειραματική Αξιολόγηση

5.1 Αλγόριθμοι αξιολόγησης

Η επιλογή των αλγορίθμων για την αξιολόγηση των συστημάτων έγινε με γνώμονα την όσο το δυνατόν πιο αντιπροσωπευτική εκτίμηση πάνω σε αυτά. Επιλέχθηκαν διαφορετικοί αλγόριθμοι καθένας από τους οποίους εκπροσωπεί ξεχωριστή κατηγορία αλγορίθμων ανάλυσης γράφων: αλγόριθμοι κεντρικότητας (centrality), αλγόριθμοι εύρεσης μονοπατιού (graph traversal) και αλγόριθμοιο ελέγχου συνεκτικότητας (connectivity). Κατά αντιστοιχία, λοιπόν, έχουμε:

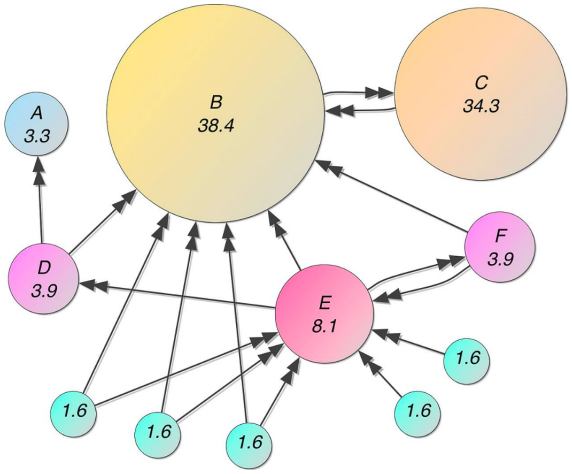
* Ο αλγόριθμος **PageRank** [34] ανιχνεύει τη "σημαντικότητα" κάθε κόμβου στο γράφο βάσει του πλήθους και της ποιότητας των ακμών - σχέσεων που συγκεντρώνει. Συγκεκριμένα, ας θεωρήσουμε έναν κόμβο A στον οποίο καταλήγουν ακμές από τους κόμβους T1, ..., Tn (π.χ οι σελίδες Ti αναφέρουν τη σελίδα A). Έπειτα ορίζουμε τη συνάρτηση $C(v)$ όπου υπολογίζει το πλήθος των εξερχόμενων ακμών από έναν κόμβο . Ορίζουμε, λοιπόν, το PageRank του κόμβου A σύμφωνα με την παρακάτω σχέση:

$$PR(A) = (1-d) + d \times (\frac{PR(T1)}{C(T1)} + ... + \frac{PR(Tn)}{C(Tn)})$$

  Το d είναι παράμετρος απόσβεσης και έρευνες δείχνουν ότι μία καλή τιμή είναι το $d = 0.85$. Διαισθητικά, ο αλγόριθμος PageRank μπορεί να θεωρηθεί ως μοντέλο της συμπεριφοράς του χρήστη. Αν θεωρήσουμε ότι ένας χρήστης επιλέγει να δει μια σελίδα τυχαία και συνεχίζει να επιλέγει συνδέσμους μέχρι να βαρεθεί και να ξαναξεκινήσει τη διαδικασία από την αρχή σε άλλη τυχαία σελίδα. Η πιθανότητα αυτός ο χρήστης να επιλέξει μία σελίδα είναι και το PageRank της. Το d, λοιπόν, αντιστοιχεί στην πιθανότητα του χρήστη να βαρεθεί μία σελίδα και να επιλέξει κάποια άλλη.

  Η υλοποιήση του αλγορίθμου στα προγραμματιστικά μοντέλα κατανεμημένων συστημάτων (BSP, SG, GAS) στηρίζεται στην παρακάτω κεντρική ιδέα. Κάθε κόμβος του γράφου, στέλνει στους γείτονές του (εξερχόμενες ακμές) την τιμή Page Rank που έχει μέχρι στιγμής. Στη συνέχεια εκτελεί τη συνάρτηση που περιγράψαμε παραπάνω και

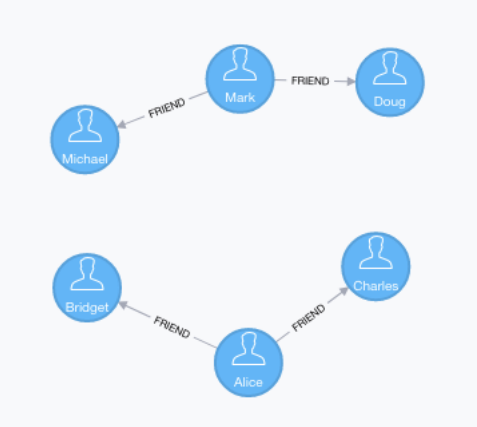
έπειτα ενημερώνει εκ νέου τους γείτονες για τη νέα του τιμή. Πρόκειται για τις πιο ¨ακριβές' υλοποιήσεις μεταξύ των τριών αλγορίθμων που θα μελετήσουμε σύμφωνα με το κόστος επικοινωνίας και επεξεργασίας που απαιτούν. Αυτό οφείλεται στο γεγονός ότι όλοι οι κόμβοι είναι ενεργοί για κάθε επανάληψη του αλγορίθμου και είναι απαραίτητο να ενημερώνουν όλους τους γείτονές τους για την ανανεωμένη τους τιμή. Στην περίπτωση του Neo4j, το Page Rank των κόμβων υπολογίζεται διασχίζοντας τυχαία τον γράφο και μετρώντας τη συχνότητα που ¨χτυπάμε' κάθε κόμβο (score). Πρόκειται για έναν αλγόριθμο υψηλών απαιτήσεων σε μνήμη.



Σχήμα 5.1: Ο αλγόριθμος PageRank. Η σημαντικότητα του κόμβου B εξαρτάται τόσο από το πλήθος των κόμβων που αναφέρουν τον κόμβο B καθώς επίσης και την ποιότητα των κόμβων που τον αναφέρουν.

* Ο αλγόριθμος **Weakly Connected Components** υπολογίζει το πλήθος των συνεκτικων συνιστωσών που υπάρχουν στον γράφο. Ως συνεκτική συνιστώσα, ορίζουμε τον υπογράφο του γράφου G που είναι μέγιστα συνδεδεμένο. Κάθε κόμβος ανήκει σε μία συνεκτική συνιστώσα. Η υλοποίηση του αλγορίθμου στα παράλληλα και κατανεμημένα συστήματα γίνεται με την εξής διαδικασία: Κάθε κόμβος ξεκινάει θέτοντας ως τιμή τον αντιπροσωπευτικό αριθμό που έχει (id) και στέλνοντας στους γείτονές του μήνυμα που περιέχει την τιμή του. Έπειτα, σε ενδιάμεσο στάδιο, για κάθε τιμή που λαμβάνεται από τα εισερχόμενα μηνύματα, ο κόμβος ελέγχει αν υπάρχει τιμή που να είναι μικρότερη από αυτήν που ήδη έχει. Αν υπάρχει, τότε αντικαθιστά την τιμή του με την ελάχιστη τιμή που έλαβε και ενημερώνει εκ νέου τους γείτονές του για την αλλαγή. Η διαδικασία αυτή τερματίζεται όταν δεν υπάρχουν άλλες ενημερώσεις να γίνουν ή όταν ο αλγόριθμος φτάσει στο μέγιστο επιτρεπτό όριο επαναλήψεων. Σε αντίθεση με την υλοποίηση στον αλγόριθμο Page Rank βλέπουμε ότι στον WCC όσο περνάνε οι επαναλήψεις τόσο μηνύματα όσο και οι ενεργοί κόμβοι μειώνονται.

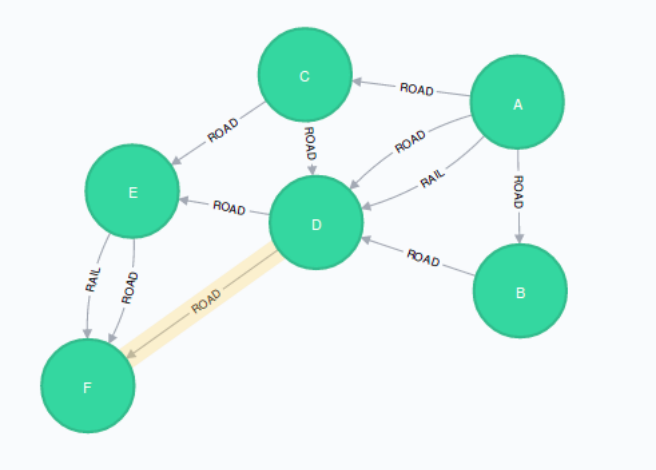
  Σχετικά με το Neo4j, η υλοποίηση στηρίζεται στην ιδέα καταμερισμού του συνόλου των κόμβων σε ανεξάρτητα συνόλα. Η δομή δεδομένων καλείται disjoint-set data structure. Στη συνέχεια μελετώντας τις ακμές ο αλγόριθμος συγχωνεύει τα σύνολα. Ο αλγόριθμος



Σχήμα 5.2: Ο αλγόριθμος Weakly Connected Components. Σε αυτό το παράδειγμα βλέπουμε ότι ο γράφος αποτελείται από δύο συνεκτικές συνιστώσες.

είναι γνωστός και ως Disjoint Set Union.

* Ο αλγόριθμος **Single Source Shortest Path** υπολογίζει την απόσταση που απέχει ένας κόμβος-αφετηρία από τους υπόλοιπους κόμβους στο γράφο. Αν δεν υπάρχει μονοπάτι που να συνδέει δύο κόμβους, τότε η απόστασή τους θεωρείται άπειρη. Σε αυτόν τον αλγόριθμο η διαδικασία ξεκινάει από έναν κόμβο v. Στην πρώτη επανάληψη ο κόμβος v ενημερώνει τους γείτονές του στέλνοντας μήνυμα έναν μετρητή αρχικοποιημένο στη μονάδα που αντιστοιχεί στο πλήθος ακμών που τους χωρίζει. Στην επόμενη επανάληψη, οι γείτονες του v θα λάβουν το μήνυμα θα ενημερώσουν την τιμή τους με τον μετρητή και θα τον προωθήσουν στους δικούς τους γείτονες αυξημένο κατά 1. Όταν τελειώσουν οι προωθήσεις των μηνυμάτων ή όταν φτάσουμε στο προκαθορισμένο πλήθος επαναλήψεων ο αλγόριθμος τερματίζει.



Σχήμα 5.3: Ο αλγόριθμος Single Source Shortest Paths. Εκκινώντας από τον κόμβο D, η απόσταση από τον κόμβο F είναι 1 ενώ από τον C άπειρο

Σε αυτόν τον αλγόριθμο η διαδικασία μηνυμάτων είναι αυξητική. Ένας κόμβος ξεκινάει

να στέλνει μηνύματα σε λίγους γείτονες και στη συνέχεια ¨πλημμυρίζει' όλος ο γράφος. Ωστόσο, είναι σημαντικό να τονίσουμε πως σε αντίθεση με τους προηγούμενους αλγορίθμους η υπολογιστική διαδικασία είναι αρκετά μικρότερη. Αυτό συμβαίνει διότι κάθε κόμβος ενεργοποιείται μόνον για τα βήματα του αλγορίθμου όπου χρειάζεται να ενημερώσει την απόσταση του σε μία μικρότερη. Αφού στείλει τα μηνύματα στους γείτονες, ο κόμβος απενεργοποιείται και δε συμμετέχει επιπλέον στην υπολογιστική διαδικασία. Όσον αφορά το Neo4j, υλοποιείται ένας δ-βηματικός αλγόριθμος [35] που υπολογίζεται ότι είναι πιο αποδοτικός από τον Dijkstra.

Οι παραπάνω αλγόριθμοι υλοποιήθηκαν και στις τέσσερις πλατφόρμες. Τα μοντέλα εκτέλεσης που ακολουθήσαμε στις κατανεμημένες πλατφόρμες συγκεντρώνονται στον Πίνακα 5.1. Όσον αφορά τους αλγορίθμους στο Neo4j, ως κεντρική βάση δεδομένων για γράφους, δεν υλοποιούν κάποιο από τα παράλληλα μοντέλα που περιγράψαμε στην Ενότητα 2. Αντίθετα, γίνονται βελτιστοποιήσεις πάνω σε γνωστούς αλγορίθμους όπως είδαμε παραπάνω.

	Υλοποιήσεις αλγορίθμων		
Συστήματα	**PageRank**	**SSSP**	**WCC**
Apache Flink Gelly	SG	GSA	GSA
Apache Giraph	BSP	BSP	BSP
Apache Spark GraphX	GAS	GAS	GAS
Neo4j	Random Traversals	Disjoint Set Union	δ-βηματικός

Πίνακας 5.1: Υλοποίηση αλγορίθμων στις πλατφόρμες επεξεργασίας

5.2 Δεδομένα

Η επιλογή των δεδομένων πάνω στο οποία θα εκτελέσουμε τα πειράματα είναι σημαντική προκειμένου τα αποτελέσματα να είναι όσο το δυνατόν αντιπροσωπευτικά. Η διαφορετικότητα των δεδομένων μπορεί να μας δώσει μία καλύτερη εικόνα για τη συμπεριφορά των συστημάτων επεξεργασίας καθώς επίσης και τα σημεία στα οποία το ένα υπερτερεί έναντι του άλλου.

Στον Πίνακα 5.2 παρουσιάζονται τα δεδομένα που χρησιμοποιήθηκαν. Επιλέξαμε να χρησιμοποιήσουμε πραγματικά σύνολα δεδομένων [39, 40, 41], καθώς επίσης να δημιουργήσουμε και τρία συνθετικά με τη βοήθεια του LDBC Datagen [39] (βλ. υποενότητα 5.2.1). Τα δεδομένα επιλέχθηκαν βάσει του μεγέθους τους αλλά και βάσει των χαρακτηριστικών που παρουσιάζουν ως γράφοι όπως η πυκνότητα, το average clustering coefficient [40] που εκφράζει την τάση του γράφου να σχηματίζει κοινότητες καθώς επίσης και το assortativity [41] το οποίο δείχνει την τάση των κόμβων να σχετίζονται (να υπάρχει ακμή) με κόμβους παρόμοιου βαθμού.

Παρακάτω βρίσκουμε ορισμένες πληροφορίες για τα σύνολα δεδομένων:

- **Amazon**: Το δίκτυο δημιουργήθηκε κατά το crawling στον ιστότοπο της Amazon. Προκείπτει από την υπηρεσία του ιστότοπου να παρέχει την πληροφορία ¨Ο πελάτης

Όνομα	**Πλήθος κόμβων**	**Πλήθος ακμών**	**Είδος**	Πυκότητα γράφου
Amazon	0.33M	0.93M	Π	1.7×10^{-5}
DBLP	0.32M	1.05M	Π	2.0×10^{-5}
Datagen-Scale 10	0.07M	1.94M	Σ	7.9×10^{-4}
Datagen-Scale 30	0.17M	6.02M	Σ	4.2×10^{-4}
Datagen-Scale 100	0.45M	19.94M	Σ	1.9×10^{-4}
Live Journal	4.85M	68.99M	Π	5.9×10^{-6}
Orkut	3.07M	117.18M	Π	2.5×10^{-5}

Πίνακας 5.2: Τα χαρακτηριστικά των δεδομένων γράφων για τα πειράματα ως προς το πλήθος ακμών. (Π): για πραγματικό σύνολο δεδομένων, (Σ): για συνθετικό.

που αγόρασε το συγκεκριμένο αντικείμενο αγόρασε επίσης ... ¨. Οι ακμές, λοιπόν, αντιστοιχούν στην σχέση των προϊόντων.

- **DBLP**: Πρόκειται για τις αναφορές μεταξύ εργασιών που βρίσκονται στην πλατφόρμα dblp και αφορούν βιβλιογραφία στην επιστήμη των υπολογιστών.
- **Live Journal**: Πρόκειται για εναν γράφο που περιλαμβάνει τους χρήστες του Live Journal καθώς και τις σχέσεις φιλίας μεταξύ τους.
- **Orkut**: Πρόκειται για εναν γράφο που περιλαμβάνει τους χρήστες του Orkut καθώς και τις σχέσεις φιλίας μεταξύ τους.

Στη συνέχεια θα μελετήσουμε γράφους βάσει του clustering coefficient. Η έννοια clustering (συσταδοποίηση) στηρίζεται στην παρατήρηση πως σχεδόν σε όλα τα δίκτυα, οι κόμβοι τείνουν να σχηματιζουν μικρές ομάδες μεταξύ έτσι ώστε πολλές ακμές να βρίσκονται μέσα στις ομάδες και λίγες μεταξύ τους. Σε ένα κοινωνικό δίκτυο, για παράδειγμα, οι άνθρωποι σχηματίζουν ομάδες όπου σχεδόν κάθε μέλος γνωρίζει το άλλο. Η έννοια της συσταδοποίησης, λοιπόν, είναι πολύ σημαντική για τους γράφους και υπάρχουν αρκετές μετρικές για να την χαρακτηρίσουν. Μία πολύ γνωστή μετρική είναι το clustering coefficient το οποίο εκφράζει την τάση του γράφου να σχηματίζει τρίγωνικές σχέσεις μεταξύ των κόμβων. Αυτή η μετρική δεν αφορά διμερείς γράφους καθώς σε τέτοιους γράφους δεν υφίστανται τρίγωνα.

Σε αυτό το στάδιο θα μελετήσουμε τη συμπεριφορά των τελεστών πάνω σε γράφους που έχουν παρόμοιο πλήθος ακμών αλλά διαφορετική τιμή στην παράμετρο clustering coefficient, χρησιμοποιώντας μόνο ρεαλιστικά δεδομένα (Πίνακας 5.3).

Όνομα	Πλήθος κόμβων	Πλήθος ακμών	C.C
YouTube	9.38M	3.22M	0.14%
Flickr	1.74M	15.55M	11.2%
Catster	0.63M	15.70 M	2.84%

Πίνακας 5.3: Τα χαρακτηριστικά των δεδομένων γράφων για τα πειράματα του clustering coefficient (C.C.).

- **YouTube**: Πρόκειται για εναν γράφο που περιλαμβάνει τους χρήστες του YouTube καθώς και τις σχέσεις φιλίας μεταξύ τους.
- **Flickr**: Αντίστοιχα με το πρώτο, αναπαρίστανται οι χρήστες του Flickr και οι σχέσεις μεταξύ τους.
- **Catster**: Σε αυτόν τον γράφο περιλαμβάνονται οι οικογενειακές σχέσεις μεταξύ κατοικίδιων σύμφωνα με συγκεκριμένες βάσεις δεδομένων.

Από την άλλη, το Assortativity ρ εκφράζεται ως μία τιμή μεταξύ του -1 και του 1. Ένας γράφος καλείται assortative [41] όταν οι κόμβοι που έχουν μεγάλο βαθμό (degree) συνδεόνται κατά μέσο όρο κόμβους υψηλού βαθμού. Σε αυτή την περίπτωση το ρ είναι θετικό. Ωστόσο, αν στον γράφο υπάρχουν πολλές σχέσεις μεταξύ κόμβων διαφορετικού βαθμού, τότε ο γράφος χαρακτηρίζεται disassortative και το ρ γίνεται αρνητικό.

Αναλυτικότερα, η έννοιτα του assortativity έχει άμεση σχέση με την ανθεκτικότητα του δικτύου ως προς τη συνεκτικότητα [42]. Σε περίπτωση αποτυχίας ενός κόμβου υψηλού βαθμού σε ένα assortative δίκτυο οι γείτονες του αποτυχημένου κόμβου, που είναι και αυτοί υψηλού βαθμού, ελαχιστοποιούν την πιθανότητα να αποσυνδεθεί το δίκτυο. Αντίθετα, σε ένα disasortative δίκτυο υπάρχει μεγαλύτερος κίνδυνος για αποσύνδεση του δικτύου καθώς πολλά μονοπάτια εξαρτώνται από κόμβους υψηλού βαθμού.

Για τα πειράματα πάνω στο assortativity χρησιμοποιήσαμε δύο πραγματικά σύνολα δεδομένων προκειμένου να μελετήσουμε την επίδραση της παραμέτρου στην επίδοση των συστημάτων. Στον Πίνακα 5.4 βλέπουμε ορισμένα στοιχεία των γράφων.

Όνομα	Πλήθος κόμβων	Πλήθος ακμών	Assortativity
Baidu - Baike	0.41M	3.30M	0.71%
Italy CNR	0.33M	3.11M	−0.19%

Πίνακας 5.4: Τα χαρακτηριστικά των δεδομένων γράφων για τα πειράματα του assortativity.

- **Baidu-Baike**: Γράφος που περιγράφει τις αναφορές μεταξύ των άρθρων που υπάρχουν στη διαδικτυακή εγκυκλοπαίδεια Baidu-Baike.
- **Italy CNR**: Δίκτυο που παρέχεται από το Ιταλικό συμβούλιο έρευνας.

Τέλος, ο γράφος Catster που είδαμε στον Πίνακα 5.3 θα χρησιμοποιηθεί και για τη μελέτη των στρατηγικών καταμερισμού (partition strategies) του γράφου στους εργάτες της συστάδας. Την πιο πλούσια βιβλιοθήκη στρατηγικών καταμερισμού την έχει το GraphX. Περιλαμβάνει: Τυχαίο καταμερισμό (Random partitioning), Τυχαίο κανονικοποιημένο καταμερισμό (Canonical random partitioning), καταμερισμό ακμών σε μία διάσταση (1D Edge Partitioning) και καταμερισμό ακμών σε δύο διαστάσεις (1D Edge Partitioning). Σε αντίθεση με άλλα συστήματα, το GraphX επιτρέπει την ανάθεση περισσότερων από έναν καταμερισμό (partition) σε κάθε εργάτη. Ιδανικά, προτείνεται η ανάθεση ενός τμήματος του καταμερισμού σε κάθε πυρήνα.

Στους τυχαίους καταμερισμούς, το GraphX αναθέτει τις ακμές στους εργάτες μέσω μίας συνάρτησης κατατεμαχισμού (hash function) πάνω στους αριθμούς ταυτότητας (IDs) του κόμβου πηγής και προορισμού της ακμής. Η διαφορά μεταξύ των δύο στρατηγικών συνίσταται στο ότι στον κανονικοποιημένο καταμερισμό οι αντίθετες ακμές (πχ. οι ακμές (u,v) και (v,u)) αντιστοιχούν στο ίδιο τμήμα καταμερισμού ενώ στον απλά τυχαίο καταμερισμό αυτό δεν εξασφαλίζεται.

Όσον αφορά τη στρατηγική του καταμερισμού ακμών σε μία διάσταση, στηρίζεται και εκείνη σε μία σύναρτηση κατατεμαχισμού αλλά αυτήν τη φορά ως όρισμα λαμβάνεται μόνον ο κόμβος πηγή. Ως αποτέλεσμα, η παρούσα στρατηγική εξασφαλίζει πως κάθε τμήμα κατακερματισμού διαθέτει όλες τις ακμές ενός κόμβου. Αυτή η στρατηγική είναι αποδοτική για κόμβους με μικρό πλήθος ακμών.

Σχήμα 5.4: Καταμερισμός ακμών σε δύο διαστάσεις: Ο κόμβος u αντιστοιχίζεται στο μηχάνημα 1 ενώ ο v στο μηχάνημα 9. Η ακμή (u, v) μπορεί να αποθηκευτεί στα μηχανήματα 7 ή 9.

Ο καταμερισμός ακμών σε δύο διαστάσεις οργανώνει τους κόμβους εργάτες σε έναν τετραγωνικό πίνακα. Κάθε κόμβος μπορεί να αντιστοιχηθεί (μέσω της hash συνάρτησης) σε ένα μηχάνημα καθώς επίσης και στα μηχανήματα που βρίσκονται στην ίδια στήλη και γραμμή. Κάθε ακμή αντιστοιχίζεται τουλάχιστον σε δύο μηχανήματα (βλ. Σχήμα 5.4). Αυτή η στρατηγική έχει ως άνω όριο επαναλήψεων των κόμβων (replication factor) στα μηχανήματα που ισούται με $2\sqrt{N} - 1$ όπου N το πλήθος των μηχανημάτων. Η μέθοδος αυτή λειτουργεί για οποιονδήποτε αριθμό N που δεν είναι πρώτος.

Εξ ορισμού κάθε σύστημα χρησιμοποιεί στρατηγική τυχαίου καταμερισμού στους εργάτες της συστάδας.

Τα δεδομένα που μελετάμε δεν ήταν από την αρχή στη μορφή που επιθυμούμε. Αντιθέτως, περιείχαν και πληροφορίες οι οποίες ήταν περιττές για το σκοπό των μετρήσεών μας. Επομένως, προβήκαμε σε μία προεπεξεργασία των δεδομένων έτσι ώστε να καταλήξουμε σε κατάλληλη τελική μορφή. Συγκεκριμένα, η είσοδος των πειραμάτων αποτελείται από δύο αρχεία. Το ένα αφορά τους χρήστες και το άλλο τις σχέσεις μεταξύ τους. Το αρχείο των χρηστών αποτελείται από δύο στήλες: στην μία υπάρχει ένας τυχαίος αριθμός που αντιστοιχεί στην ταυτότητα του κόμβου και στη δεύτερη μία συμβολοσειρά που αφορά το όνομά του. Από την άλλη, το αρχείο των σχέσεων μπορεί να έχει δύο τύπους λόγω του τρόπου που διαβάζουν τα δεδομένα τα συστήματα επεξεργασίας. Συγκεκριμένα, όπως βλέπουμε στο Σχήμα 5.5 είτε οι σχέσεις θα εκφράζονται μία σε κάθε γραμμή είτε θα δίνονται σε μορφή πίνακα γειτνίασης.

```
<IDx> <IDy>                    <src_1> <dst_2> .. <dst_N>
   ...                                ...
<IDz> <IDw>                    <src_N> <dst_2> .. <dst_N>
```

Σχήμα 5.5: Διαφορετικές μορφοποιήσεις του αρχείου ακμών.

5.2.1 Παραγωγή συνθετικών δεδομένων

Η παραγωγή συνθετικών δεδομένων παρέχει τη δυνατότητα να ελέγξουμε γράφους των οποίων τα χαρακτηριστικά είναι δύσκολο να βρεθούν σε πραγματικά δεδομένα. Επίσης, μας δίνουν τη δυνατότα να μελετήσουμε τα όρια ενός συστήματος. Υπάρχουν πολλά εργαλεία τα οποία παράγουν συνθετικούς γράφους. Χρησιμοποιήσαμε το LDBC Social Network Benchmark Data generator - Datagen και πρόκειται για μία κλιμακώσιμη, συνθετική γεννήτρια κοινωνικών δικτύων. Ο λόγος που την επιλέξαμε είναι επειδή οι γράφοι που παράγει διατηρούν πολλά από τα χαρακτηριστικά των ρεαλιστικών γράφων: συσχέτιση δεδομένων, άνιση κατανομή των ακμών στους κόμβους. Τα δεδομένα που παράγονται μιμούνται την δραστηριότητα ενός κοινωνικού δικτύου για ένα χρονικό διάστημα (σχέσεις, μηνύματα, αναρτήσεις, σχόλια κλπ.). Εξ ορισμού η κατανομή των σχέσεων που δημιουργούνται μιμείται αυτήν του γράφου του Facebook, ωστόσο υπάρχει ένα σύνολο από μοντέλα που μπορούν να επιλεχθούν

Η γεννήτρια υλοποιείται με το προγραμματιστικό πλαίσιο Map Reduce συνεπώς λειτουργεί πάνω σε περιβάλλον Hadoop. Για την παραγωγή των δικών μας δεδομένων χρησιμοποιήσαμε την γεννήτρια ldbc snb datagen v0.2.7 πάνω σε Hadoop 2.6.0.

5.3 Περιβάλλον αξιολόγησης

Τα πειράματα διεξάγονται πάνω σε κατανεμημένο περιβάλλον και συγκεκριμένα σε μία συστάδα από πέντε εικονικές μηχανές Openstack δομής του εργαστηρίου μας. Το ένα από

τα πέντε μηχανήματα χρησιμοποιείται αποκλειστικά για τη λειτουργία του Neo4j Community edition v3.3.5. Τα υπόλοιπα μηχανήματα σχηματίζουν μία συστάδα που λειτουργεί πάνω από δομή Hadoop v2.4.1 με συνολική χωρητικότητα δίσκου 30GB. Όσον αφορά τη μνήμη το μηχάνημα που λειτουργεί το Neo4j έχει συνολικά 4GB ενώ για τα κατανεμημένα συστήματα διαθέστουμε 4GB ανά εργάτη. Προκειμένου να παρακολουθήσουμε τη συμπεριφορά και την κατάσταση κάθε κόμβου, εγκαταστήσαμε το Ganglia Monitoring System v3.6.0 ένα κλιμακώσιμο, κατανεμημένο σύστημα παρακολούθησης υπολογιστικών συστημάτων. Στον κόμβο-αφέντη της συστάδας Hadoop τρέχει επίσης και το IReS προκειμένου να μπορεί να διαχειριστεί τους πόρους της συστάδας. Μετά από την εκτέλεση κάθε τελεστή, το IReS συγκεντρώνει τις πληροφορίες που έλαβε από το ganglia και τις αποθηκεύει σε μία βάση MongoDB με σκοπό τον χαρακτηρισμό (profiling) του τελεστή αυτού. Επομένως, στον κόμβο αφέντη λειτουργεί και βάση MongoDB v4.0.2. Τέλος, όσον αφορά τα συστήματα επεξεργασίας γράφων χρησιμοποιήσαμε τις εξής εκδόσεις: Spark v2.1.3, Flink v1.4.2, Giraph v1.1.0.

5.4 Πειραματική διαδικασία

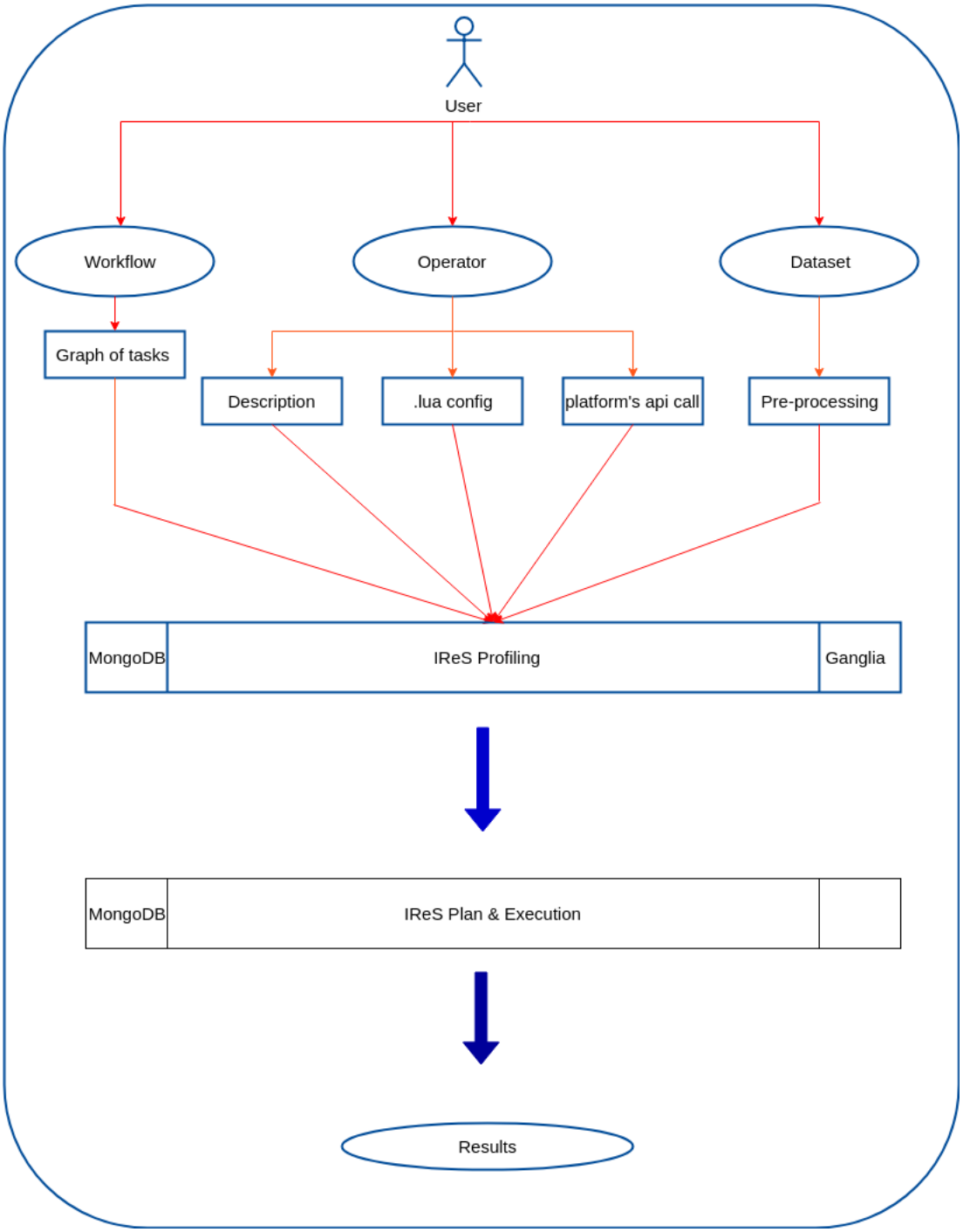
Όπως έχουμε αναφέρει, σκοπός της διπλωματικής είναι τόσο η αξιολόγηση των συστημάτων επεξεργασίας γράφων όσο και η μελέτη της επίδοσης του συστήματος IReS. Συνεπώς, τα πειράματά μας αποτελούνται από δύο στάδια. Στο Σχήμα 5.6 βλέπουμε τη διαδικασία που ακολουθούμε προκειμένου να διεξάγουμε πειράματα. Αφού προεπεξεργαστούμε τα δεδομένα (βλ. Ενότητα 5.2), πρέπει να ορίσουμε τόσο τις ροές εργασίας όσο και τους τελεστές που θέλουμε να υλοποιήσουμε.

Συγκεκριμένα, στις ροές εργασίας, ορίζουμε το γράφο υποεργασιών που εκτελούνται δηλώνοντας τη ροή των δεδομένων από τη μία υποεργασία στην άλλη. Όσον αφορά τον ορισμό των τελεστών τα πράγματα είναι πιο αυστηρά. Για να μπορέσει το IReS να αναγνωρίσει έναν τελεστή σε ένα σύστημα επεξεργασίας, ο χρήστης καλείται να του ορίσει τρία αρχεία:

- Το αρχείο περιγραφής: Σε αυτό το αρχείο ο χρήστης καθορίζει τυχόν περιορισμούς ως προς τα δεδομένα (σύστημα αρχείων που αποθηκεύονται, τύπος αρχείου), την εκτέλεση (σύστημα επεξεργασίας) και τη βελτιστοποίηση (είσοδος, μετρικές, πολιτική, κόστος).

- Το script αρχείο: Σε αυτό το αρχείο περιγράφουμε στον IReS πως θα εκτελούσαμε τον τελεστή αν τον τρέχαμε απευθείας από το σύστημα επεξεργασίας. Προσδιορίζουμε λοιπόν την API κλήση που θα αναθέσει στο σύστημα την εκτέλεση του τελεστή.

- Το αρχείο ⟨ Όνομα τελεστή ⟩.lua : Στην Ενότητα 4 χαρακτηρίσαμε το IReS ως ένα σύστημα που βασίζεται πάνω στο YARN. Για την εκτέλεση κάθε υποεργασίας, το IReS, μέσω του YARN, δημιουργεί containers τα οποία εκτελούν το σύνολο του κώδικα που έχει οριστεί στο προηγούμενο αρχείο. Όλες οι ρυθμίσεις αποθηκεύονται σε ένα αρχείο με κατάληξη .lua και αφορούν το Cloudera Kitten.



Σχήμα 5.6: Η ροή διεξαγωγής των πειραμάτων

5.4.1 Οργάνωση πειραμάτων

Τα πειράματα χωρίζονται σε δύο μεγάλες κατηγορίες. Όπως ήδη αναφέραμε, η πρώτη κατηγορία αφορά τη σύγκριση των συστημάτων επεξεργασίας. Για αυτό το σκοπό θα γίνει εκτενής ανάλυση της επίδοσης των τεσσάρων συστημάτων πάνω στις εξής μετρικές: χρόνος εκτέλεσης, χρήση δικτύου και χρήση CPU. Πάνω σε κάθε σύστημα επεξεργασίας θα τρέξουμε

και τους τρεις αλγορίθμους πέντε φορές για κάθε αρχείο εισόδου και θα παραστήσουμε την median τιμή. Παράλληλα, όλες οι τιμές θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου στον IReS που θα προβλέπει την βέλτιστη λύση. Όσον αφορά την εκπαίδευση του μοντέλου, ως χαρακτηριστικό θεωρούμε το πλήθος των ακμών |E| που υπάρχουν στον γράφο ενώ η έξοδος είναι ο χρόνος εκτέλεσης.

Στη συνέχεια θα μελετήσουμε την επίδοση των συστημάτων βασιζόμενοι σε χαρακτηριστικά της δομής του γράφου. Συγκεκριμένα, θα λάβουμε μετρήσεις σε γράφους παρόμοιου μεγέθους αλλά διαφορετικής σύνθεσης. Προς αυτό το σκοπό θα πάρουμε μετρήσεις για χρόνο εκτέλεσης βάσει του clustering coefficient και του assortativity. Και σε αυτό το σημείο θα κάνουμε από πέντε μετρήσεις σε κάθε αρχείο εισόδου και θα κρατήσουμε τον μεδιαν.

Τέλος, αφού γίνει ο χαρακτηρισμός κάθε τελεστή κάθε συστήματος στον IReS, θα προσπαθήσουμε να εκτελέσουμε μία σύνθετη ροή εργασίας. Αρχικά, θα αφήσουμε το σύστημα να προτείνει τους τελεστές που κρίνει πιο αποδοτικούς (βάσει χρόνου εκτέλεσης). Ύστερα, θα τρέξουμε τη ροή εργασίας με κάθε σύστημα ξεχωριστά και θα αναλύσουμε την επιλογή του IReS.

5.5 Σύγκριση συστημάτων επεξεργασίας γράφων

5.5.1 Χρόνος εκτέλεσης

Στα Σχήματα 5.7, 5.9, 5.8 βλέπουμε την επίδοση που έχουν τα συστήματα επεξεργασίας ως προς το χρόνο εκτέλεσης. Οι μετρήσεις έγιναν στα επτά πρώτα σύνολα δεδομένων που αναφέραμε στην Ενότητα 5.2 (Πίνακας 5.2) και πάνω στους αλγορίθμους Page Rank, Connected Components και Shortest Paths. Τέλος, στα σχήματα 5.11 έως 5.13 βλέπουμε την επίδοση του εκάστοτε συστήματος εκφρασμένη σε πλήθος κόμβων/ακμών που μπορεί καθένα να επεξεργαστεί ανά δευτερόλεπτο. Βασικές παρατηρήσεις επί των αποτελεσμάτων είναι:

- Όσο οι γράφοι μπορούν να χωρέσουν στη μνήμη ενός μηχανήματος, το Neo4j αποτελεί την καλύτερη επιλογή για όλους τους αλγορίθμους. Για τον αλγόριθμο Page Rank βλέπουμε πως όσο το πλήθος των ακμών αυξάνεται και μπορεί να χωρέσει στη μνήμη, το Neo4j έχει όλο και καλύτερη επίδοση όντας $x2.8$ φορές ταχύτερο από το Spark, $x4.6$ από το Giraph και $x5.8$ φορές από το Flink. Παρατηρούμε, επίσης, πως στο Neo4j ο τελεστής Single Source Shortest Paths πετυχαίνει το μικρότερο κέρδος απέναντι στις κατανεμημένες πλατφόρμες σε σχέση με το κέρδος του στους άλλους δύο τελεστές. Τέλος, μπορεί κανείς να παρατηρήσει το πόσο δραματικά πέφτει η επίδοση του Neo4j για το τελευταίο σύνολο δεδομένων. Στο Σχήμα 5.10 παρουσιάζεται ο χρόνος που χρειάζεται το σύστημα να φορτώσει τον γράφο στη βάση. Αυτό συμβαίνει καθώς η μνήμη του μηχανήματος δεν επαρκεί ώστε να φορτωθεί η βάση αποδοτικά. Όταν πλέον έχει φορτωθεί, η διαθέσιμη μνήμη για επεξεργασία καθυστερεί δραματικά την εκτέλεση των τελεστών.

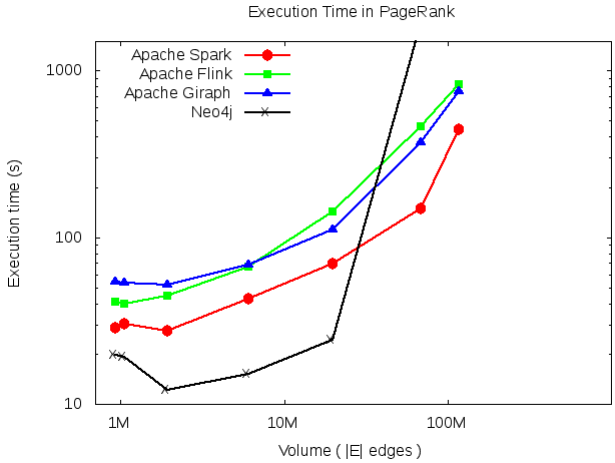
- Όσον αφορά τη σύγκριση μεταξύ των κατανεμημένων συστημάτων το Apache Spark (Gelly) πετυχαίνει ως επί τον πλείστον την καλύτερη επίδοση έναντι των υπολοίπων.

Συγκεκριμένα, στον τελεστή Page Rank αποτελεί την καλύτερη επιλογή ανεξαρτήτως γράφου, πετυχαίνοντας επίδοση ακόμη και $x3$ έναντι του Apache Flink Gelly.

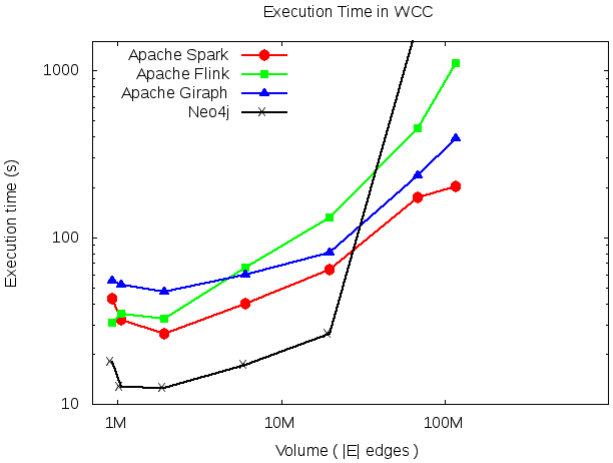
- Επιπλέον, αξιοσημείωτη είναι η δυνατότητα κλιμακωσιμότητας του Apache Giraph έναντι των υπόλοιπων συστημάτων στους τελεστές πλην του Page Rank. Ιδιαίτερα, στον αλγόριθμο SSSP το Giraph είναι $x2.7$ φορές ταχύτερο από το Spark. Το γεγονός αυτό μπορεί να εξηγηθεί κοιτάζοντας τον αλγόριθμο του SSSP ο οποίος μπορεί να υλοποιηθεί βέλτιστα στο μοντέλο BSP του Giraph. Το γεγονός πως το Giraph κατανέμει τους κόμβους στους εργάτες και όχι τις ακμές σε συνδυασμό με τη φύση του προγραμματιστικού μοντέλου BSP αποδίδουν καλύτερα σε αλγόριθμους που εστιάζουν σε υποσύνολο του γράφου. Αντίθετα, η απαίτηση του τελεστή Page Rank για ταυτόχρονη επεξεργασία σε όλοκληρο το γράφο καθιστά την επίδοση του Giraph τη χειρότερη μεταξύ των συστημάτων. Βασική αιτία αποτελεί η ανάγκη συγχρονισμού σε κάθε υπερβήμα σε συνδυασμό με τον καταμερισμό του γράφου στους εργάτες. Πιο αναλυτικά, οι κόμβοι του γράφου διαμοιράζονται στους εργάτες. Είναι πολύ πιθανόν ένας εργάτης να επεξεργάζεται πιο 'δημοφιλείς' κόμβους σε σχέση με κάποιον άλλον με αποτέλεσμα να υπάρχει καθυστέρηση μεταξύ των υπερβημάτων.

- Και στους 3 τελεστές παρατηρούμε πως αν και το Giraph παρουσιάζει κλιμακωσιμότητα, για μικρούς γράφους η απόδοση δεν είναι και τόσο καλή. Αυτό οφείλεται στο γεγονός πως το Giraph, ως hadoop εργασία, χρησιμοποιεί τον διαχειριστή πόρων του Hadoop Map Reduce ο οποίος για μικρούς γράφους επιφέρει σημαντικό overhead στην απόδοση.

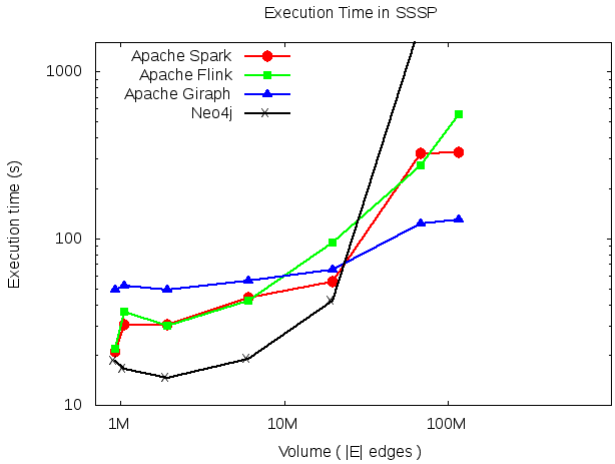
- Τέλος, βλέπουμε πως το Apache Flink (Gelly) για τα μικρά και μεσαία μεγέθη πετυχαίνει παραπλήσια επίδοση με τα υπόλοιπα συστήματα χωρίς όμως να ξεχωρίσει σε κάποιο σημείο. Το Apache Flink αδυνατεί να ακολουθήσει την επίδοση του Apache Spark για έναν βασικό λόγους. Το Apache Flink πρόκειται για πλατφόρμα επεξεργασίας ροής δεδομένων (stream processing). Συνεπώς αντιμετωπίζει, όπως περιγράψαμε και στην Ενότητα 3, την batch επεξεργασία ως υποπερίπτωση της επεξεργασίας ροής με αποτέλεσμα να μην έχει βελτιστοποιηθεί για τέτοιου είδους εργασίες. Επίσης, αν και το Apache Flink Gelly στοχεύει στην επεξεργασία δεδομένων σε πραγματικό χρόνο και εκεί έχει βελτιστοποιηθεί, οι μετρήσεις δείχνουν πως μπορεί να ανταγωνιστεί και τα υπόλοιπα συστήματα.



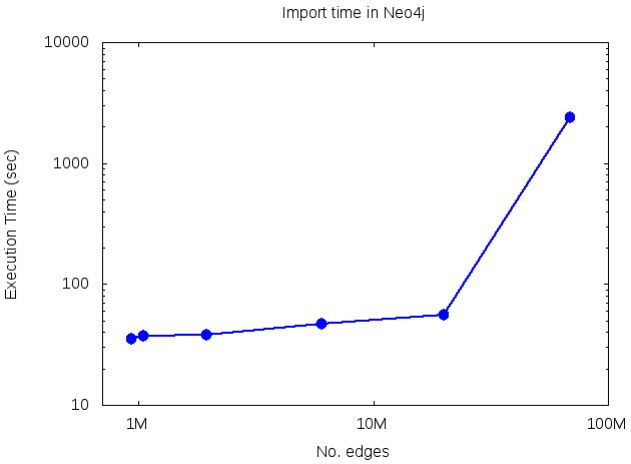
Σχήμα 5.7: Χρόνος εκτέλεσης του τελεστή Page Rank



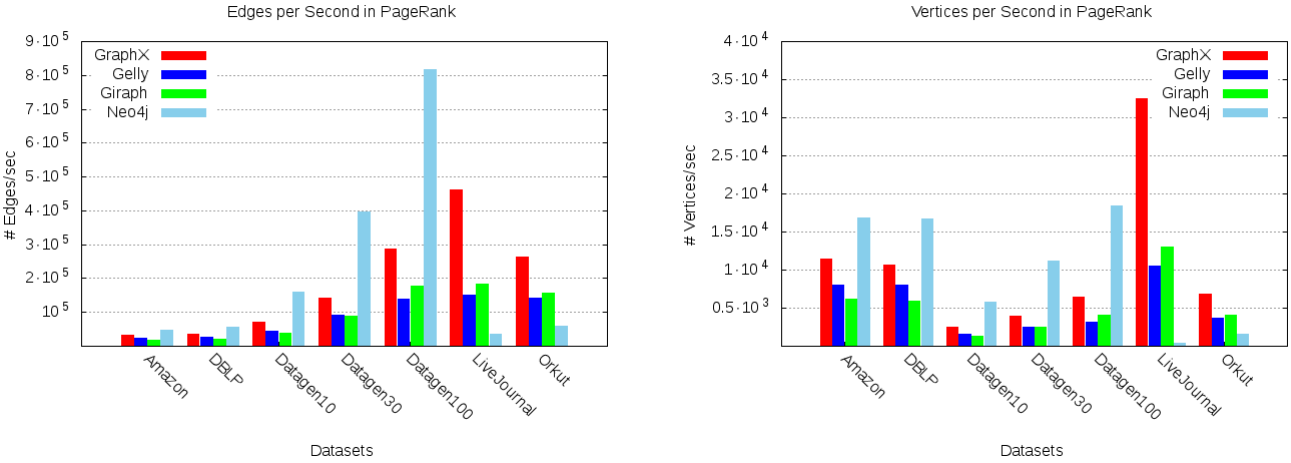
Σχήμα 5.8: Χρόνος εκτέλεσης του τελεστή WCC



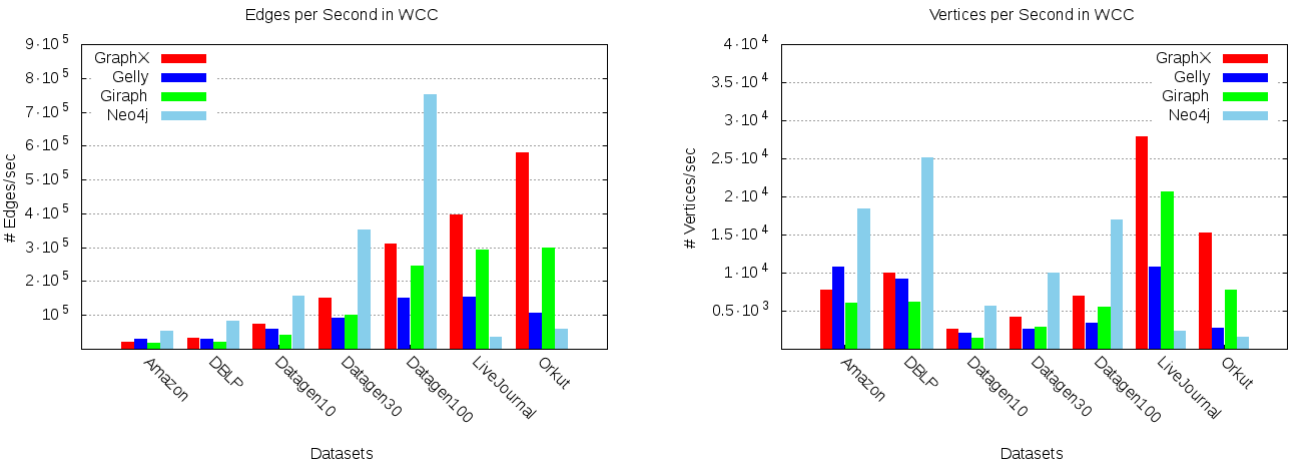
Σχήμα 5.9: Χρόνος εκτέλεσης του τελεστή SSSP



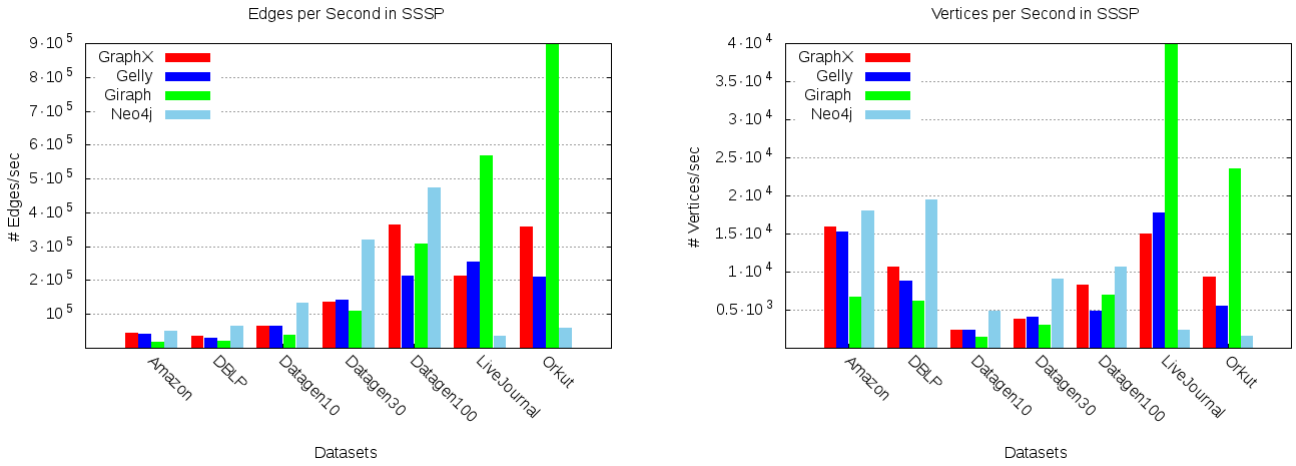
Σχήμα 5.10: Χρόνος φόρτωσης του γράφου στο Neo4j



Σχήμα 5.11: Επίδοση στον τελεστή PageRank



Σχήμα 5.12: Επίδοση στον τελεστή WCC

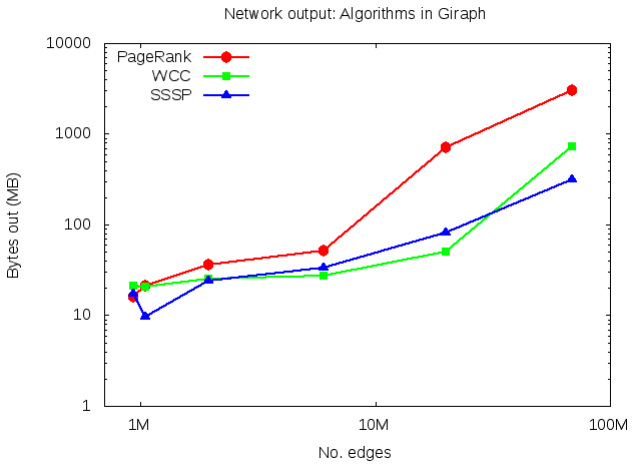


Σχήμα 5.13: Επίδοση στον τελεστή SSSP

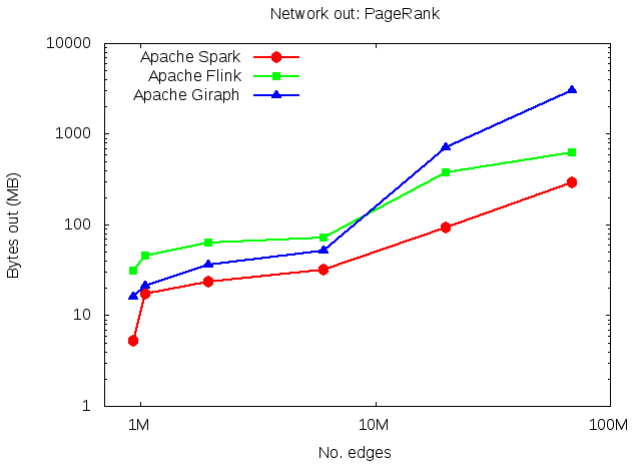
5.5.2 Χρήσιμοποίηση Δικτύου

Αρχικά, στο Σχήμα 5.14 μπορούμε να αποκτήσουμε μία διαίσθηση των απαιτήσεων κάθε τελεστή για επικοινωνία. Βλέπουμε πως το Page Rank είναι εμφανώς πιο απαιτητικό σε επικοινωνία καθώς το περιεχόμενο του μηνύματος είναι μεγαλύτερο σε σχέση με των άλλων τελεστών. Στο Page Rank στέλνουμε στους γειτονικούς κόμβους έναν αριθμό διπλής ακρίβειας ένω στο SSSP έναν μετρητή από το ένα έως το δέκα και στο WCC τον αριθμό id του κάθε κόμβου. Αφετέρου, όπως εξηγήσαμε και στην υποενότητα 5.2.1 ο τελεστής Page Rank οφείλει να ενημερώνει κάθε γείτονά του για την νέα του τιμή με αποτέλεσμα να παράγεται μεγάλος όγκος μηνυμάτων. Συνεπώς, η χρήση του δικτύου αναμένεται μεγαλύτερη.

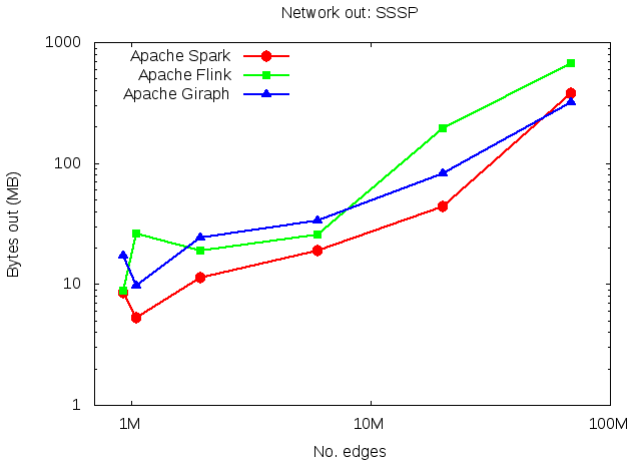
Στα Σχήματα 5.15, 5.16, 5.17 βλέπουμε την χρήση του δικτύου που κάνει κάθε σύστημα σε κάθε τελεστή. Οι μετρήσεις έγιναν πάνω στα δεδομένα του Πίνακα 5.2. Σε αυτά τα πειράματα έχουμε αποκλείσει το Neo4j καθώς πρόκειται για κεντρικό σύστημα. Παρατηρούμε, λοιπόν, πως το Apache Spark GraphX πετυχαίνει την ελάχιστη χρήση του δικτύου ανεξαρτήτως τελεστή. Αυτό συμβαίνει λόγω της εκτεταμένης βελτιστοποίησης που έχει επιτευχθεί στο Apache Spark τόσο στη διατήρηση (persist) των RDD στους εργάτες όσο και κατά την ανάμειξη (shuffling) των δεδομένων. Επιπροσθέτως, βλέπουμε πως το Apache Giraph είναι το αμέσως επόμενο οικονομικό σύστημα για τους τελεστές πλην του Page Rank.



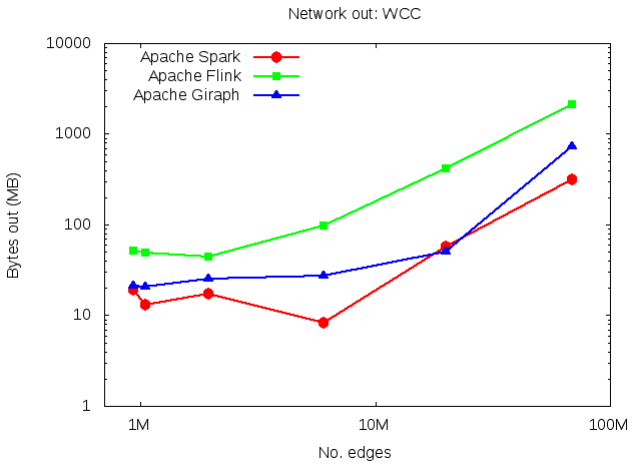
Σχήμα 5.14: Η χρήση δικτύου των τελεστών στους Giraph



Σχήμα 5.15: Η χρήση δικτύου κατά την εκτέλεση του τελεστή Page Rank



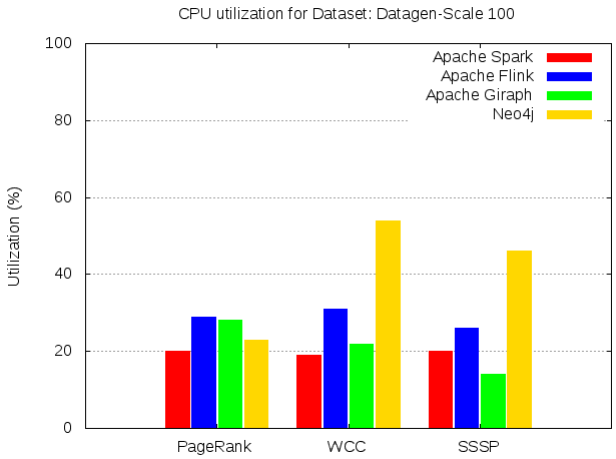
Σχήμα 5.16: Η χρήση δικτύου κατά την εκτέλεση του τελεστή SSSP



Σχήμα 5.17: Η χρήση δικτύου κατά την εκτέλεση του τελεστή WCC

5.5.3 Χρήσισμοποίηση Κεντρικής Μονάδας Επεξεργασίας

Στο Σχήμα 5.18 βλέπουμε την μέγιστη τιμή που φτάνει η χρήση της Κεντρικής Μονάδας Επεξεργασίας των τεσσάρων συστημάτων για το σύνολο δεδομένων Datagen-Scale 100. Συγκεκριμένα, για τα κατανεμημένα συστήματα πήραμε το μέσω όρο των μέγιστων τιμών που παρατηρήθηκε σε ολόκληρη τη hadoop συστάδα ενώ για το Neo4j πήραμε τη μέγιστη τιμή χρήση της ΚΜΕ του ενός μηχανήματος.



Σχήμα 5.18: Η χρήση δικτύου κατά την εκτέλεση του τελεστή CPU

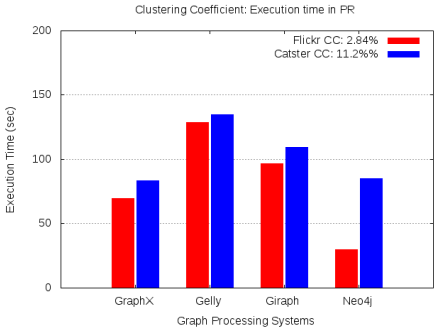
5.5.4 Clustering Coefficient και Πυκνότητα

Στον πίνακα 5.3 βλέπουμε τα χαρακτηριστικά των γράφων που θα εξετάσουμε. Θα μελετήσουμε τόσο την επίδραση της πυκνότητας όσο και του clustering coefficient. Συγκεκριμένα στα Σχήματα 5.19 - 5.22 βλέπουμε:

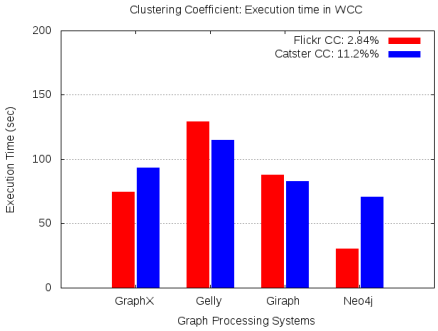
- Όσον αφορά τη μετρική clustering coefficient [40] θα εστιάσουμε στους γράφους Flickr και Catster (Πίνακας 5.3) για κάθε τελεστή καθώς έχουν παρεμφερή πυκνότητα. Εκεί μπορούμε να δούμε πως στον τελεστή WCC το Giraph πετυχαίνει καλύτερα αποτελέσματα έναντι των υπόλοιπων κατανεμημένων συστημάτων για μεγαλύτερό c.c.

- Επιπροσθέτως, όσον αφορά τον τελεστή SSSP βλέπουμε πως, ανάμεσα στους δύο γράφους που αναφέραμε, όσο μεγαλύτερο είναι το c.c. τόσο χειρότερη είναι και η επίδοση. Αυτό μπορεί να εξηγηθεί με την παρατήρηση ότι όσο περισσότερες τριγωνικές σχέσεις υπάρχουν τόσο περισσότερα μηνύματα (και κατέπέκταση ανάγκη για επεξεργασία) θα απαιτούνται προκειμένου να επικοινωνήσουν οι γείτονες μεταξύ τους. Τα μηνύματα αυτά είναι άσκοπα καθώς από δεδομένη κορυφή ενός τριγώνου οι δύο γείτονες απέχουν μονάδα και είναι άσκοπο να επιχειρήσουν να στείλουν μεταξύ τους μήνυμα προτείνωντας μικρότερη απόσταση.

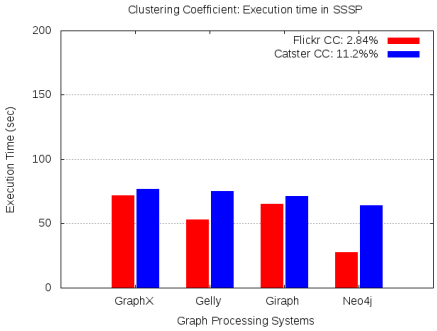
- Τέλος, πρόκειται να αναφέρουμε μία σημαντική παρατήρηση σχετικά με την επίδοση των συστημάτων ως προς την πυκνότητα του γράφου εισόδου. Υπενθυμίζουμε πως



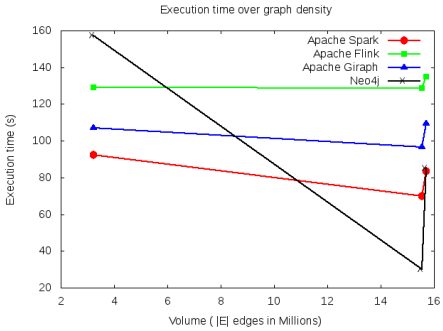
Σχήμα 5.19: Clustering Coefficient στους τελεστές Page Rank



Σχήμα 5.20: Clustering Coefficient στους τελεστές WCC



Σχήμα 5.21: Clustering Coefficient στο SSSP

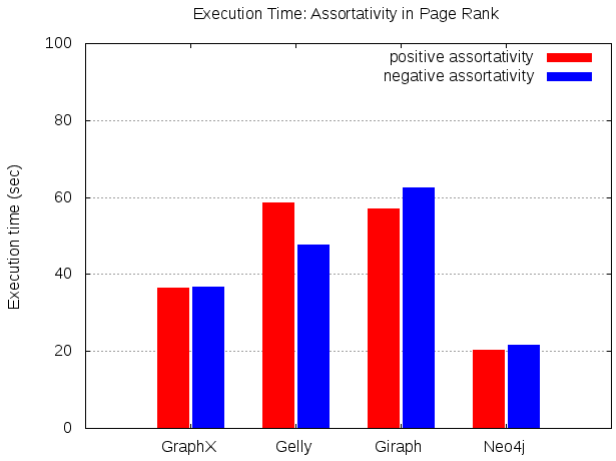


Σχήμα 5.22: Χρόνος εκτέλεσης βάσει πυκνότητας γράφου στο Page Rank

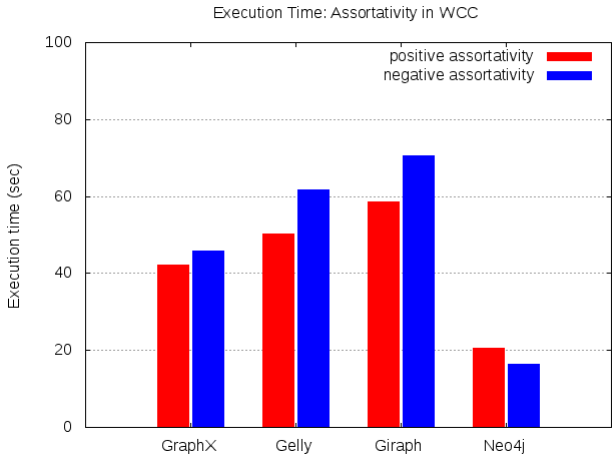
ο γράφος YouTube έχει υποτριπλάσιο πλήθος ακμών προς επεξεργασία ωστόσο καταφέρνει να πετύχει παρόμοιο, πολλές φορές και χειρότερο, χρόνο από τους υπόλοιπους γράφους. Αυτό οφείλεται στο πόσο αραιός είναι ο γράφος του YouTube σε σχέση με τους υπόλοιπους. Επιπλέον το γεγονός ότι οι τελεστές πλην του SSSP λειτουργούν σε ολόκληρο τον γράφο συνεπάγεται το ότι πρέπει κάθε κόμβος να υπολογίζει την τιμή του. Συνεπώς, αν και λιγότερες οι ακμές, το πλήθος των κόμβων και κατεπέκταση η πυκνότητα του γράφου μπορεί να επηρεάσει επικίνδυνα την επίδοση του συστήματος.

5.5.5 Assortativity

Στα Σχήματα 5.23 έως 5.28 παρουσιάζουμε τις μετρήσεις που λάβαμε πάνω στους γράφους Baidu-Baike και Italy-CNR τόσο για το χρόνο εκτέλεσης όσο και για τη χρήση του δικτύου. Σύμφωνα με τα σχήματα λαμβάνουμε τις εξής σημαντικές παρατηρήσεις:



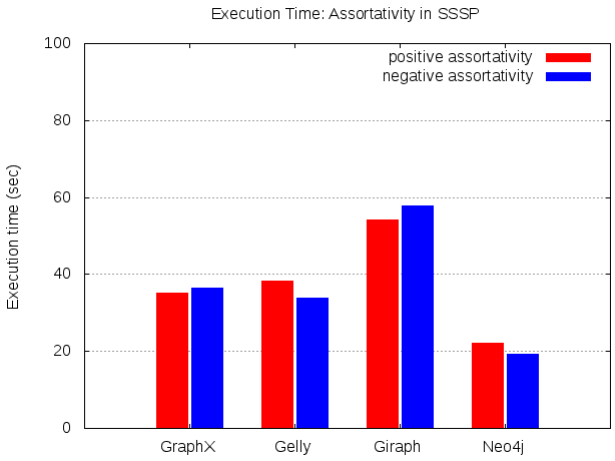
Σχήμα 5.23: Ο χρόνος εκτέλεσης των συστημάτων για διαφορετικά assortativity: Page Rank



Σχήμα 5.24: Ο χρόνος εκτέλεσης των συστημάτων για διαφορετικά assortativity: WCC

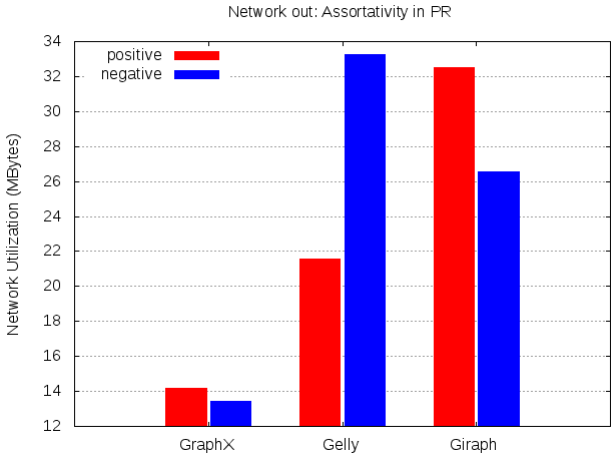
- Από την μελέτη της επίδοσης των συστημάτων στο χρόνο εκτέλεσης για μεταβαλλόμενο assortativity (πέρα από την υπέροχη του Neo4j), δεν προκύπτει μία επικρατούσα κατανεμημένη πλατφόρμα επεξεργασίας. Το Apache Spark GraphX είναι καλύτερο στους τελεστές Page Rank και WCC ανεξαρτήτως assortativity ενώ στο SSSP παρατηρούνται διακυμάνσεις.

- Όσον αφορά τα μηνύματα τα οποία εξέρχονται συνολικά από κάθε εργάτη παρατηρούμε πως δεν υπάρχει μία πλατφόρμα που να υπερτερεί έναντι των υπολοίπων. Συγκεκριμένα,

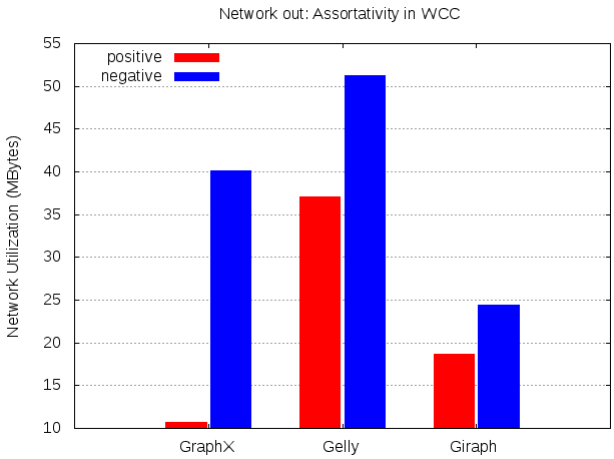


Σχήμα 5.25: Ο χρόνος εκτέλεσης των συστημάτων για διαφορετικά assortativity: SSSP

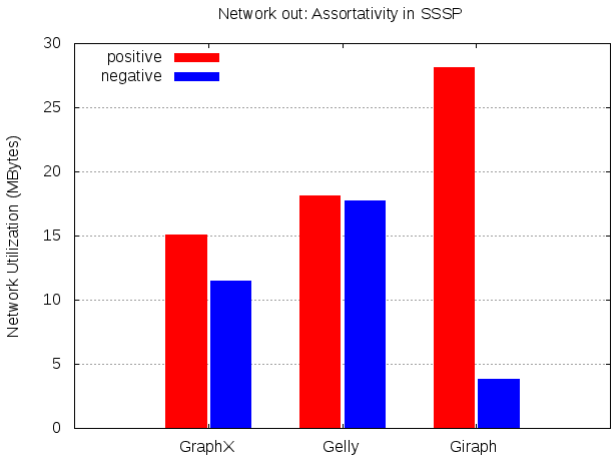
βλέπουμε πως καμία πλατφόρμα δεν υπερτερεί σε όλους τους τελεστές για δεδομένο assortativity. Αν και στο Page Rank δεν παρατηρείται μεγαλη απόκλιση στην επίδοση των συστημάτων για διαφορετικό assortativity, στους άλλους δύο τελεστές οι διαφορές είναι εμφανής. Χαρακτηριστική είναι η συμπεριφορά του Giraph όπου για αρνητικό assortativity αποδεικνύεται καλύτερη επιλογή από του GraphX για τον τελεστή WCC. Μια πιθανή εξήγηγη μπορεί να δοθεί στην περίπτωση του Giraph, στο SSSP όπου ο κόμβος-αφετηρία έχει μεγάλο πλήθος εξερχόμενων ακμών. Επιλέξαμε τον κόμβο-αφετηρία έτσι ώστε να έχει γείτονες με υψηλό βαθμό γεγονός που καθιστά την ανάγκη επικοινωνίας αυξανόμενη με το πέρασμα των επαναλήψεων. Αντιθέτως, στο αρνητικό assortativity ο κόμβος αφετηρία, έχει περισσότερες πιθανότητες να επικοινωνήσει με κόμβους με λιγότερες ακμές και επομένως με λιγότερες ανάγκες για επικοινωνία. Η περιγραφή αυτή αντικατοπτρίζεται και στην εικόνα 5.28.



Σχήμα 5.26: Η χρήση του δικτύου για διαφορετικά assortativity: Page Rank



Σχήμα 5.27: Η χρήση του δικτύου για διαφορετικά assortativity: WCC

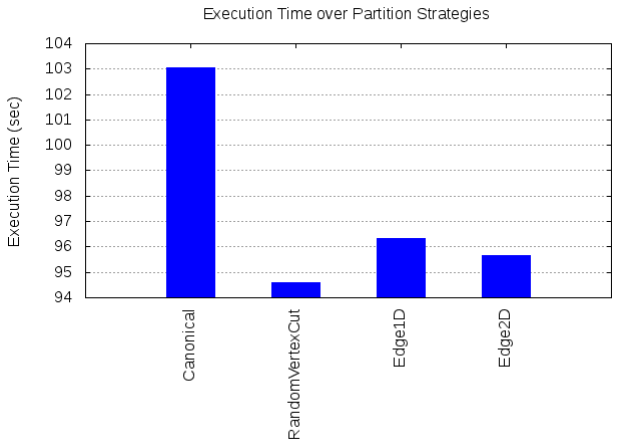


Σχήμα 5.28: Η χρήση του δικτύου για διαφορετικά assortativity: SSSP

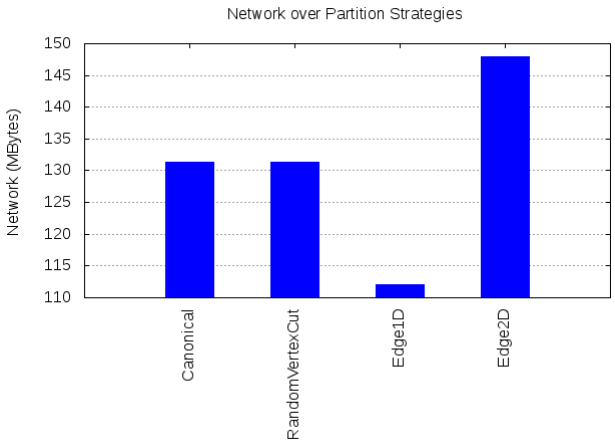
5.5.6 Στρατηγικές καταμερισμού γράφων

Σε αυτήν την ενότητα θα μελετήσουμε διαφορετικές στρατηγικές καταμερισμού γράφων πάνω στο γράφο datagen-100. Στα Σχήματα 5.29, 5.30 βλέπουμε το χρόνο εκτέλεσης και τη χρήση δικτύου του συστήματος Apache Spark GraphX ως προς τις στρατηγικές καταμερισμού πάνω στον αλγόριθμο PageRank.

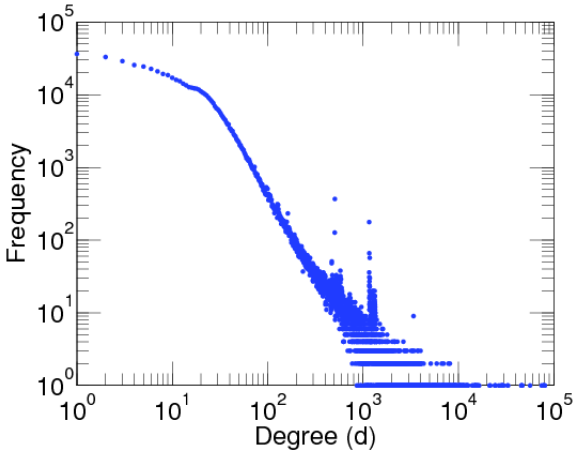
Όλα τα δεδομένα που μελετάμε έχουν ένα κοινό χαρακτηριστικό. Η κατανομή των ακμών στους κόμβους ακολουθεί την κατανομή νόμου δύναμης (power-law distribution). Στο Σχήμα 5.31 βλέπουμε την κατανομή για το σύνολο δεδομένων που μελετάμε. Για τέτοιου είδους γράφους, όπου η πλειονότητα των κόμβων δεν έχει μικρό βαθμό (οι ακμές είναι πολύ περισσότερες από μερικές δεκάδες) προτείνεται η χρήση των στρατηγικών καταμερισμού ακμών (1D και 2D)[43]. Αυτό συμβαίνει καθώς μέσω αυτών των στρατηγικών τίθεται ανώ όριο στον παράγοντα αντιγραφής (replication factor) των κόμβων στους εργάτες. Ως αποτέλεσμα, κρατώντας τον παράγοντα αντιγραφής χαμηλό κρατάμε χαμηλά και τον χρόνο εκτέλεσης.



Σχήμα 5.29: Ο χρόνος εκτέλεσης του GraphX για διαφορετικές στρατηγικές καταμερισμού.



Σχήμα 5.30: Η χρήση του δικτύου του GraphX για διαφορετικές στρατηγικές καταμερισμού.



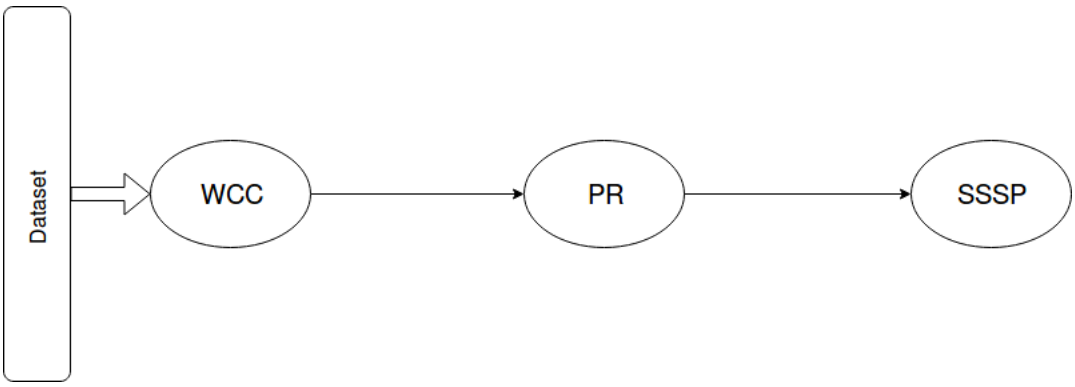
Σχήμα 5.31: Η κατανομή του γράφου Catster ακολουθεί την κατανομή νόμου δύναμης.

5.6 Αξιολόγηση συστήματος IReS

Στην προηγούμενη υποενότητα παρουσιάσαμε τα αποτελέσματα που δείχνουν την επίδοση κάθε συστήματος ξεχωριστά πάνω σε κάθε τελεστή. Σε αυτό το σημείο θα μελετήσουμε την επίδοση του IReS πάνω σε μία πιο σύνθετη ροή εργασίας και στο αν μπορεί να προτείνει κάποια καλύτερη υλοποίηση από το να υλοποιούσαμε τη ροή εργασίας με ένα μόνο σύστημα επεξεργασίας.

5.6.1 Παρουσίαση ροής εργασίας

Για τον σκοπό αυτό θα κατασκευάσουμε μία ροή εργασίας που να περιλαμβάνει και τους τρεις τελεστές που μελετήσαμε. Στο Σχήμα 5.32 βλέπουμε την ροή των εργασιών που καλούνται να υλοποιηθούν. Πρόκειται για έναν γράφο εργασιών και η φιλοσοφία του είναι η εξής:

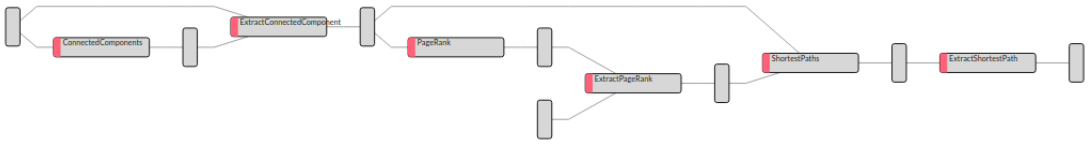


Σχήμα 5.32: Η υπο εξέταση ροή εργασίας.

- Αρχικά, φιλτράρουμε τα δεδομένα έτσι ώστε να έχουμε έναν μεγάλο συνεκτικό γράφο, όπου κάθε κόμβος συνδέεται με τους υπόλοιπους. Χρησιμοποιούμε, λοιπόν, τον αλγόριθμο Weakly Connected Components και κρατάμε τις ακμές που συνδέοουν τους κόμβους που ανήκουν στην πιο μεγάλη συνεκτική συνιστώσα.
- Στη συνέχεια βρίσκουμε τον σημαντικότερο κόμβο του γράφου μέσω του τελεστή PageRank.
- Τέλος, ξεκινώντας από τον πιο σημαντικό κόμβο (σύμφωνα με το PageRank) υπολογίζουμε όλα τα ελάχιστα μονοπάτια του γράφου κρατώντας τους κόμβους που απέχουν το πολύ 10 βήματα μακριά.

5.6.2 Αποτελέσματα: Χρόνος εκτέλεσης

Σε αυτήν την υποενότητα μελετάμε η επίδοση τεσσάρων υλοποιήσεων της ροής εργασίας που περιγράψαμε στην Ενότητα 5.6.1. Οι πρώτες τρεις υλοποιήσεις αφορούν την εκτέλεση της



Σχήμα 5.33: Η ροή εργασίας όπως φαίνεται από το γραφικό περιβάλλον του IReS. Τα ενδιάμεσα βήματα ,ανάμεσα στους τελεστές, μετατρέπουν τα δεδομένα στην κατάλληλη μορφή για το επόμενο στάδιο.

ροής εργασίας χρησιμοποιώντας ξεχωριστά τις κατανεμημένες πλατφόρμες. Η τελευταία αφορά την πρόταση που το σύστημα IReS θεωρεί βέλτιστη. Μετρήσεις για την πλατφόρμα Neo4j δεν παρουσιάζονται καθώς η επίδοσή της είναι χαμηλή. Αυτό οφείλεται στην ροή εργασίας που εκτελούμε. Αρχικά, στο στήσιμο της πειραματικής διαδικασίας κάνουμε την παραδοχή πως τα αρχεία εισόδου (ο γράφος δεδομένων) βρίσκονται μόνον στο Hadoop σύστημα αρχείων ενώ δεν είναι φορτωμένα στην πλατφόρμα Neo4j. Επιπλέον, για τη δεδομένη ροή εργασίας, το Neo4j θα πρέπει να φορτώσει δεδομένα στη βάση της δύο φορές καθιστώντας την επιλογή του σε κάποιον συνδυασμό απαγορευτική. Συνεπώς, η πλατφόρμα δεν συμμετέχει στα παρακάτω πειράματα.

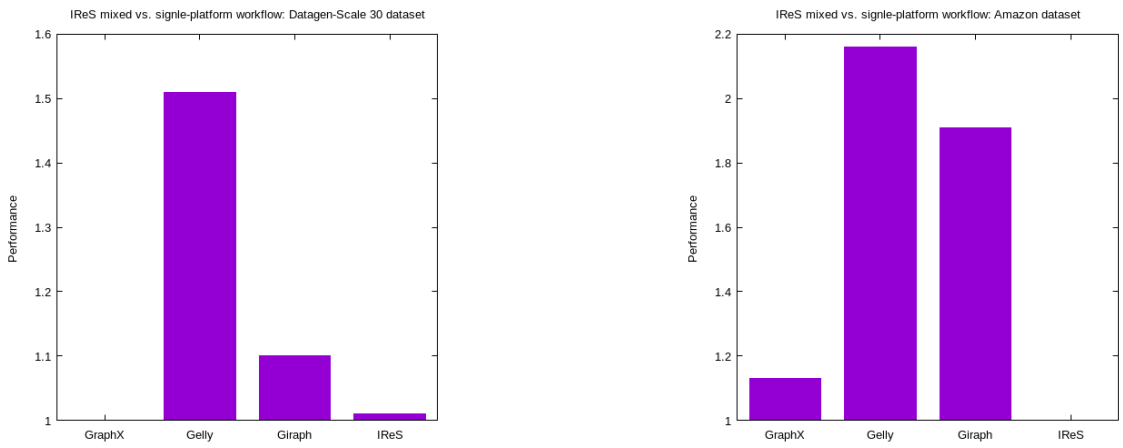
Στα Σχήματα 5.34 βλέπουμε την επίδοση ως προς το χρόνο εκτέλεσης των υλοποιήσεων που περιγράψαμε. Τα αποτελέσματα συνοψίζονται ως εξής:

- Κατανοώντας κανείς το όφελος του πολυσυστηματικού περιβάλλοντος και της λειτουργίας του IReS μπορεί να βγάλει ένα βασικό συμπέρασμα. Για κάθε τελεστή σε μία ροή εργασίας, το IReS επιλέγει το βέλτιστο, σύμφωνα με το μοντέλο που έχει επιλέξει, σύστημα προς υλοποίηση. Συνεπώς, για δεδομένη ροή εργασίας, το IReS πετυχαίνει πάντα καλύτερη απόδοση έναντι των ροών εργασίας ενός συστήματος. Η μόνη εξαίρεση βρίσκεται όταν το IReS προτείνει να εκτελεστεί η ροή εργασίας χρησιμοποιώντας ένα σύστημα.

- Στο πρώτο διάγραμμα του σχήματος 5.34 συμβαίνει ακριβώς το αυτό που περιγράψαμε παραπάνω. Το IReS προτείνει υλοποίηση χρησιμοποιώντας τελεστές Apache Spark. Ως εκ τούτου, αναμένουμε η επίδοσή τους να είναι παρεμφερής.

- Στο δεύτερο διάγραμμα βλέπουμε την υπεροχή της πρότασης του IReS έναντι οποιασδήποτε απλής λύσης ενός συστήματος. Συγκεκριμένα, το σύστημα προτείνει να υλοποιηθεί η ροή εργασίας χρησιμοποιώντας Apache Spark πλην του τελεστή WCC όπου (βλ. Σχήμα 5.8) χρησιμοποιείται Apache Flink. Σε αυτήν την περίπτωση επιτυγχάνεται απόδοση κοντά στο $x1.2$ έναντι του Spark.

- Επιπλέον, μπορούμε να σημειώσουμε την κλιμακωσιμότητα που παρουσιάζει το Giraph. Όσο μεγαλώνει το πλήθος των ακμών στο γράφο, τόσο η επίδοση του Giraph προσεγγίζει τη βέλτιστη λύση του IReS.



Σχήμα 5.34: Επίδοση του IReS για διαφορετικούς γράφους

- Βλέπουμε λοιπόν πως η επιλογή του IReS θα είναι πάντα καλύτερη, στη χειρότερη παρεμφερής, έναντι οποιασδήποτε απλής λύσης. Συγκεκριμένα, για το γράφο με πλήθος ακμών 68M του Πίνακα 5.2 μπορούμε να περιμένουμε την επίδοση του IReS να είναι ξ1.5 φορές μεγαλύτερη από την επίδοση του Spark.

- Τέλος, αξίζει να αναφέρουμε πως υπάρχει και η περίπτωση το σύστημα IReS να μην προτείνει τη βέλτιστη ροή εργασίας. Ο λόγος για τον οποίο μπορεί κάτι τέτοιο να συμβεί, αφορά το μοντέλο που επιλέγει το IReS για να επιλέξει τα συστήματα επεξεργασίας. Ωστόσο, όσο καλύτερα τροφοδοτήσουμε το IReS κατά τη φάση της εκπαίδευσης των μοντέλων (δηλ. όσες περισσότερες φορές εκτελέσουμε κάθε τελεστή για κάθε σύστημα), τόσο μικρότερη είναι η πιθανότητα να πετύχουμε τη συγκεκριμένη περίπτωση.

5.7 Σύνοψη συμπερασμάτων αξιολόγησης

Συνοψίζοντας τα σημαντικότερα σημεία κατά τη διάρκεια των πειραματικών αποτελεσμάτων, καταλήγουμε στις εξής παρατηρήσεις:

- Σύμφωνα με τα διαγράμματα, καταλήγουμε πως δεν υπάρχει μία επικρατούσα πλατφόρμα επεξεργασίας γράφων για κάθε χαρακτηριστικό των γράφων και για κάθε παράμετρο βελτιστοποίησης.

- Αν και επεξεργασία γράφων μεγάλης κλίμακας, η κεντρική πλατφόρμα Neo4j αποτελεί τη βέλτιστη επιλογή για οποιονδήποτε γράφο, αρκεί να χωράει στη μνήμη ενός μηχανήματος. Συγκεκριμένα πετυχαίνει επιδόσεις έως και $x2.8$.

- Οι υλοποιήσεις σε Apache Spark GraphX παρουσιάζουν την μεγαλύτερη σταθερότητα ως προς την επίδοση και σε χρόνο εκτέλεσης αλλά και σε χρήση δικτύου αγγίζοντας έως και $x2.5$ φορές μεγαλύτερη επίδοση έναντι των υπόλοιπων κατανεμημένων συστημάτων.

- Το Apache Giraph κλιμακώνει ικανοποιητικά καταφέρνοντας να ξεπεράσει το Apache Spark για τον τελεστή SSSP.
- Για το σύνολο των δεδομένων που παρουσιάσαμε, η ροή εργασίας που προτείνει το σύστημα IReS πετυχαίνει έως και $x1.5$ καλύτερη επίδοση στο χρόνο εκτέλεσης έναντι των ροών εργασίας που υλοποιούνται με ένα σύστημα.

Κεφάλαιο 6

Επίλογος

Σε αυτό το κεφάλαιο συγκεντρώνουμε τα συμπεράσματα και τις παρατηρήσεις που προέκυψαν από την πειραματική διαδικασία στην Ενότητα 5. Τέλος, αναφέρουμε το πεδίο έρευνας που αφήνει αυτή η διπλωματική και τις βελτιώσεις που μπορούν να επιτευχθούν.

6.1 Σύνοψη

Με τη διαρκή αύξηση της μοντελοποίησης των δεδομένων μεγάλης κλίμακας στη μορφή γράφων δημιουργήθηκαν πολλοί περιορισμοί και εμπόδια στην επεξεργασία και ανάλυση των δεδομένων. Ως αποτέλεσμα, δημιουργήθηκε πληθώρα συστημάτων προκειμένου να καλύψουν το πλήθος των διαφορετικών εφαρμογών που προέκυπταν. Η επιλογή του κατάλληλου συστήματος για την βέλτιστη επεξεργασία των γράφων καθίσταται απαιτικό στάδιο στην ανάπτυξη μίας εφαρμογής μιας και η απόδοση εξαρτάται τόσο από τη μορφή των γράφων και τη φύση του αλγορίθμου.

Αυτή η δυσκολία επιλογής οδήγησε στη δημιουργία συστημάτων (πχ. cloud συστήματα [31, 32, 33]) τα οποία μπορούν να υποστηρίζουν πολλές πλατφόρμες επεξεργασίας. Τέτοια πολυ-συστηματικά περιβάλλοντα απαιτούν την ύπαρξη ενός δρομολογητή ο οποίος θα συντονίζει και θα επιλέγει το καλύτερο σύστημα επεξεργασίας γράφων ανάλογα με τις δεδομένες συνθήκες. Ένας τέτοιος δρομολογητής είναι και ο IReS.

Σε αυτήν τη διπλωματική παρουσιάζουμε μία εκτενή έρευνα πάνω στις πιο γνωστές πλατφόρμες επεξεργασίας γράφων (κατανεμημένες και μη): το Apache Spark GraphX, το Apache Flink Gelly, το Apache Giraph και το Neo4j. Τα πειράματα έγιναν πάνω σε πλήθος δεδομένων προκειμένου να μελετηθεί η συμπεριφορά των συστημάτων τόσο ως προς το πλήθος των ακμών στο γράφο όσο και ως προς δομικά χαρακτηριστικά του ίδιου του γράφου όπως το clustering coefficient και το assortativity. Τα αποτελέσματα των πειραμάτων δείχνουν πως το Apache Spark είναι σταθερά πιο καλό για μεγάλο πλήθος δεδομένων. Ωστόσο, όσο αυξάνεται το πλήθος των ακμών το Apache Giraph παρουσιάζει καλύτερη κλιμακωσιμότητα για ορισμένους τελεστές πετυχαίνωντας και καλύτερη επίδοση από το Apache Spark.

Παράλληλα, με τις παραπάνω μετρήσεις εκπαιδεύσαμε τον IReS έτσι ώστε για δεδομένο πλήθος ακμών του γράφου να μπορεί να προτείνει την καλύτερη διαθέσιμη πλατφόρμα. Τα

πειράματα έγιναν πάνω στις πλατφόρμες που αναφέραμε αναθέτοντας αυτή τη φορά μία πιο σύνθετη εργασία. Εκτελώντας την εργασία για διάφορα σύνολα δεδομένων παρατηρήσαμε πως το IReS πετυχαίνει καλύτερες επιδόσεις (στη χειρότερη παρόμοιες) έναντι των υπόλοιπων ροών εργασίας.

Συνοψίζοντας, η βελτίωση που προσφέρει ένα περιβάλλον πολλαπλών συστημάτων επεξεργασίας γράφων είναι εμφανής και μπορεί να προσφέρει σημαντικά οφέλη στην επεξεργασία γράφων μεγάλης κλίμακας.

6.2 Μελλοντικές επεκτάσεις

Παρατηρώντας την επίδοση του IReS μπορούμε να κατανοήσουμε τις δυνατότητες που μπορεί να προσφέρει στην επεξεργασία γράφων και γενικότερα στην ανάλυση δεδομένων και υλοποιηση σύγχρονων εφαρμογών. Η παρούσα διπλωματική μπορεί να αποτελέσει αφετηρία για τη δημιουργία ενός συστήματος σύγκρισης και αξιολόγησης πλατφόρμων επεξεργασίας γράφων ανάλογα με τους περιορισμούς που επιθυμεί να επιβάλλει ο χρήστης. Μπορούν να τεθούν περισσότερα κριτήρια από το πλήθος των ακμών έτσι ώστε η σύγκριση να είναι όσο το δυνατόν πιο αντιπροσωπευτική. Με αυτόν τον τρόπο το IReS θα μπορέσει να αποτελέσει και ένα εργαλείο σύγκρισης συστημάτων επεξεργασίας γράφων καθώς επίσης και να εκτελεί τις ροές εργασιών που υποβάλλει ο χρήστης.

Βιβλιογραφία

[1] T. Mills, "Big Data Is Changing The Way People Live Their Lives" *in Forbes Magazine 2018*.

[2] A. Ching, S. Edunov, M.Kabiljo, D.Logothetis, S.Muthukrishnan. "One Trillion Edges: Graph Processing at Facebook Scale". *Proceedings of the VLDB Endowment*, 8(12), 1804-1815, 2015.

[3] J. Dean and S. Ghemawat. "MapReduce: Simplified Data Processing on Large Clusters". *Communications of the ACM*, 51(1), 107-113, 2008.

[4] S. Sakrm, A. Liu, and A.G. Fayoumi. "The family of mapreduce and large-scale data processing systems". *ACM Computing Surveys*, 46(1), 11, 2013.

[5] J. Lin, M. Schatz. "Design Patterns for Efficient Graph Algorithms in MapReduce". *in Proc. 8th Workshop on Mining and Learning with Graphs*, 2010.

[6] M. Capotă, T. Hegeman, A. Iosup, A. Prat-Pérez, O. Erling, P. Boncz. "Graphalytics: A Big Data Benchmark for GraphProcessing Platforms". *Proceedings of the GRADE-S'15*, 7, 2015.

[7] A. Iosup, T. Hegeman, W.L. Ngai, S. Heldens, A. Prat-Pérez, T. Manhardto, H. Chafio, M. Capotă, N. Sundaram, M. Anderson, I.G. Tănase, Y. Xia, L. Nai, P. Boncz. "LDBC graphalytics: a benchmark for large-scale graph analysis on parallel and distributed platforms". *Proceedings of the VLDB Endowment*, 9(13), 1317-1328, 2016.

[8] O. Batarfi, R. Elshawi, A. Fayoumi, R. Nouri, S. Beheshti, A. Barnawi, S. Sakr. "Large scale graph processing systems: survey and an experimental evaluation". *Cluster Computing*, 18(3), 1189-1213, 2015.

[9] M. Junghanns, A. Petermann, M. Neumann, E. Rahm. "Management and Analysis of Big Graph Data: Current Systems and Open Challenges". *Handbook of Big Data Technologies*, 14, 457-505, 2017.

[10] S. Batra, C. Tyagi. "Comparative Analysis of Relational And Graph Databases". *International Journal of Soft Computing and Engineering (IJSCE)*, 2(2), 2231-2307, 2012.

[11] D. Tsoumakos and C. Mantas. "The Case for Multi-Engine Data Analytics". *In Euro-Par 2013: Parallel Processing Workshops*, 7(4), 406-415, 2014.

[12] K. Doka, N. Papailiou, V. Giannakouris, D. Tsoumakos N. Koziris. "Mix 'n' Match Multi-Engine Analytics". *2016 IEEE International Conference on Big Data*, 2016.

[13] Apache Hadoop: *http://hadoop.apache.org*.

[14] L.G. Valiant. "A bridging model for parallel computation". *CACM*, 33(8), 1990.

[15] J.E. Gonzalez et al. "Powergraph: Distributed graph-parallel computation on natural graphs".*In: Proc. OSDI*, 2012.

[16] A. Roy et al. "Chaos: Scale-out graph processing from secondary storage".*Proceedings of the 25th Symposium on Operating Systems Principles*, 2015.

[17] Y. Low et al. "Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud".*PVLDB*, 5(8), 2012.

[18] A. Kyrola, G. Blelloch, C. Guestrin, . "GraphChi: Large-Scale Graph Computation on Just a PC".*In: Proc. OSDI*, 2012.

[19] P. Stutz, A. Bernstein, W. Cohen. "Signal/collect: Graph algorithms for the (semantic) web". *Proceedings of the ISWC*, 2010.

[20] Apache Spark: *http://spark.apache.org*.

[21] Apache Flink: *http://flink.apache.org*.

[22] Apache Giraph: *http://giraph.apache.org*.

[23] Neo4j: *http://neo4j.org/*.

[24] J.E. Gonzalez, R.S. Xin, A. Dave, D. Crankshaw, M.J. Franklin, I. Stoica. "Graphx: Graph processing in a distributed dataflow framework". *In Conference on Operating Systems Design and Implementation*, 2014 .

[25] P. Carbone, G. Fóra, S. Ewen, S. Haridi, K. Tzoumas. "Lightweight Asynchronous Snapshots for Distributed Dataflows". *In Conference on Distributed, Parallel, and Cluster Computing*, 2015 .

[26] G. Malewicz et al. "Pregel: A System for Large-scale Graph Processing". *In: Proc. SIGMOD*, 2010 .

[27] V.K. Vavilapalli et al. "Apache hadoop yarn: Yet another resource negotiator". *in Proceedings of the 4th annual Symposium on Cloud Computing*, 5, 2013 .

[28] Apache Zookeeper: *https://zookeeper.apache.org/*.

[29] K. Doka, N. Papailiou, D. Tsoumakos, C. Mantas, N. Koziris. "IReS: Intelligent, Multi-Engine Resource Scheduler for Big Data Analytics Workflows". *In: Proc. of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015 .

[30] Cloudera Kitten: *https://github.com/cloudera/kitten*.

[31] Cloudera Distribution CDH 5.2.0.: *https://www.cloudera.com/downloads/cdh/5-2-0.html*.

[32] Hortonworks Sandbox 2.1.: *http://hortonworks.com/products/hortonworks-sandbox/*.

[33] Running Databases on AWS.: *http://aws.amazon.com/running databases/*.

[34] L. Page, S. Brin, R. Motwani, T. Winograd. "The PageRank Citation Ranking: Bringing Order to the Web", 1999 .

[35] M. Kranjčević, D. Palossi, S. Pintarelli . "Parallel Delta-Stepping Algorithm for Shared Memory Architectures", *Tech. Rep. arxiv/1604.02113*, 2016 .

[36] SNAP - Standford Network Analysis Project: *http://snap.stanford.edu*.

[37] Network Repository: *http://networkrepository.com*.

[38] KONECT - The Koblenz Network Collection: *http://konect.uni-koblenz.de/*.

[39] LDBC Datagen: *http://ldbcouncil.org/blog/datagen-data-generation-social-network-benchmark*.

[40] S. Wasserman, K. Faust. "Social Network Analysis: Methods and Applications",*Cambridge: Cambridge University Press.* 1994 .

[41] M. Newman. "Assortative mixing in networks",*Phys. Rev. Lett.*, 89(20), 208701, 2002 .

[42] R. Noldus, P.V. Mieghem. "Assortativity in Complex Networks",*Journal of Computer Networks*, 2015 .

[43] S. Verma1, L.M. Leslie, Y. Shin, I. Gupta1. "An Experimental Comparison of Partitioning Strategies in Distributed Graph Processing",*Proc. VLDB Endow.* , 2017 .