ΕΠΙΛΕΓΜΕΝΕΣ

ΠΤΥΧΙΑΚΕΣ & ΔΙΠΛΩΜΑΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Τόμος 12

ΑΘΗΝΑ 2015

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

ΕΠΙΛΕΓΜΕΝΕΣ

ΠΤΥΧΙΑΚΕΣ & ΔΙΠΛΩΜΑΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Τόμος 12

Εκδίδεται μία φορά το χρόνο από το:

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών,
Πανεπιστημιούπολη, 15784 Αθήνα

Επιμέλεια έκδοσης:

Επιτροπή Ερευνητικών και Αναπτυξιακών Δραστηριοτήτων

Θ. Θεοχάρης (υπεύθυνος έκδοσης), Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Η. Μανωλάκος, Αναπληρωτής Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Γραφιστική επιμέλεια - Επιμέλεια κειμένων:

Λ. Χαλάτση, Γραφείο Προβολής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών

ISSN 1792-8826

Εξώφυλλο: Από το Διεθνές Λογοτεχνικό Φεστιβάλ «Poetry on the Road», 2001 δημιουργία των Friederike Lambers, Boris Müller, Florian Pfeffer.

Η εικόνα προκύπτει από ειδικό λογισμικό οπτικοποίησης κειμένου, στην συγκεκριμένη περίπτωση ποιήματος του φεστιβάλ.

Περιεχόμενα

Πρόλογος

Ο τόμος αυτός περιλαμβάνει περιλήψεις επιλεγμένων διπλωματικών και πτυχιακών εργασιών που εκπονήθηκαν στο Τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών κατά το διάστημα **01/01/2014 - 31/12/2014**. Πρόκειται για τον **12ο τόμο** στη σειρά αυτή. Στόχος του θεσμού είναι η ενθάρρυνση της δημιουργικής προσπάθειας και η προβολή των πρωτότυπων εργασιών των φοιτητών του Τμήματος.

Η έκδοση αυτή είναι ψηφιακή και έχει δικό της ISSN. Αναρτάται στην επίσημη ιστοσελίδα του Τμήματος και έτσι, εκτός από τη μείωση της δαπάνης κατά την τρέχουσα περίοδο οικονομικής κρίσης, έχει και μεγαλύτερη προσβασιμότητα. Για το στόχο αυτό, σημαντική ήταν η συμβολή της Λήδας Χαλάτση που επιμελήθηκε και φέτος την ψηφιακή έκδοση και πέτυχε μια ελκυστική ποιότητα παρουσίασης, ενώ βελτίωσε και την ομοιογένεια των κειμένων.

Η στάθμη των επιλεγμένων εργασιών είναι υψηλή και κάποιες από αυτές έχουν είτε δημοσιευθεί είτε υποβληθεί για δημοσίευση.

Θα θέλαμε να ευχαριστήσουμε τους φοιτητές για το χρόνο που αφιέρωσαν για να παρουσιάσουν τη δουλειά τους στα πλαίσια αυτού του θεσμού και να τους συγχαρούμε για την ποιότητα των εργασιών τους. Ελπίζουμε η διαδικασία αυτή να προσέφερε και στους ίδιους μια εμπειρία που θα τους βοηθήσει στη συνέχεια των σπουδών τους ή της επαγγελματικής τους σταδιοδρομίας.

Η Επιτροπή Ερευνητικών και Αναπτυξιακών Δραστηριοτήτων

Θ. Θεοχάρης (υπεύθυνος έκδοσης), Η. Μανωλάκος

Αθήνα, Ιούνιος 2015

ΠΤΥΧΙΑΚΕΣ ΕΡΓΑΣΙΕΣ

Υπολογισμοί Μονοπατιών σε Γράφους

Κωνσταντίνος Ε. Γιαννούσης (kgiann@di.uoa.gr)
Έκτωρ Γ. Βρεττάκης (e.vrettakis@di.uoa.gr)

Περίληψη

Σε αυτή τη πτυχιακή εργασία, υλοποιήσαμε ένα σύστημα το οποίο χειρίζεται βάσεις δεδομένων γράφων. Πιο συγκεκριμένα, εστιάζουμε στην επίλυση δυο βασικών προβλημάτων στη διαχείριση των βάσεων δεδομένων γράφων. Το ένα είναι η εύρεση του βέλτιστου τρόπου διάσχισης του γράφου. Το έτερο είναι πως να χειριστούμε και να παρουσιάσουμε τα ανακτηθέντα δεδομένα. Το σύστημα επιλέγει μονοπάτια τα οποία πληρούν μια σειρά κριτηρίων και επιτρέπει να εφαρμόζονται κριτήρια χρηστών έτσι ώστε σε κάθε επερώτηση να επιστρέφεται ένα αποτέλεσμα το οποίο να έχει νόημα για την κάθε χρήση.

Στην προσπάθειά μας για εκτενή κάλυψη υλοποιήσαμε δυο συστήματα που προσεγγίζουν το πρόβλημα με δυο ξεχωριστές τεχνικές. Στην πρώτη οι διαδρομές χαρτογραφούνται πριν από οποιαδήποτε επερώτηση ενώ στη δεύτερη τα μονοπάτια δημιουργούνται με τη χρήση τόσο υπολογισμών αλλά και συγχρονισμών. Κατόπιν, παρουσιάζουμε τις συγκρίσεις των δυο συστημάτων βγάζοντας και τα τελικά μας συμπεράσματα.

Λέξεις κλειδιά: Κατανεμημένοι Υπολογισμοί, Αλγόριθμοι Γράφων, Βάσεις Δεδομένων Γράφων, Υπολογισμοί Μονοπατιών, Εξατομίκευση Πληροφοριών

Επιβλέποντες

Ιωάννης Ιωαννίδης, Καθηγητής, Εμμανουήλ Καρβούνης, Υποψήφιος Διδάκτωρ

1. Εισαγωγή

Η ιδέα του θέματος δόθηκε με αφορμή την ανάγκη υλοποίησης μιας διεπαφής του, υπό σχεδιασμό ακόμα κατά τη συγγραφή του κειμένου, συστήματος παροχής εξατομικευμένων πληροφοριών (PAROS).

Το σύστημα αυτό έχει σκοπό να παρέχει εξατομικεύσεις, συστάσεις και άλλες υπηρεσίες προσαρμογής στα συστήματα παροχής πληροφοριών. Έτσι, η ιδέα του πειραματισμού επάνω σε κάτι τόσο εξελιγμένο ήταν μια πρόκληση για εμάς, ειδικά αν ληφθούν υπόψη οι απαιτήσεις και ο όγκος που μπορεί ένα τέτοιο σύστημα να λάβει.

Οι βάσεις δεδομένων γράφων απαντούν στις τάσεις των επιχειρήσεων σήμερα: χρήση κατά το μέγιστο πλεονέκτημα των σύνθετων και δυναμικών σχέσεων σε δεδομένα υψηλών διασυνδέσεων. Οι βάσεις δεδομένων γράφων θεωρούνται η λύση και οι νέες προσεγγίσεις προσπαθούν να παρέχουν επαρκεί μοντέλα για μεγάλους γράφους δεδομένων ειδικά σε κοινωνικούς γράφους (Social graphs) και στο γράφο του Ιστού (Web graph).

Από τις αρχές του 1990 υπήρξε ένας αριθμός μοντέλων βάσεων δεδομένων γράφων. Δυστυχώς, το ενδιαφέρον για τα μοντέλα αυτά γρήγορα παρήλθε λόγο της επερχόμενης κυριαρχίας του XML και του Internet. Σήμερα όμως, λόγω της μαζικής απήχησης των κοινωνικών δικτύων οι βάσεις δεδομένων γράφων τελούν υπό αναθεώρηση. Επιπλέον, αποτελεί γεγονός ότι μεγάλες εταιρίες όπως η Facebook, η Google και η Twitter έχουν επικεντρώσει την ανάπτυξη των επιχειρηματικών τους μοντέλων γύρω από τεχνολογίες γράφων. Όλα αυτά συντέλεσαν την εισαγωγή των βάσεων δεδομένων γράφων στο τεχνολογικό ορίζοντα.

Στο τεχνικό κομμάτι της πτυχιακής δημιουργήσαμε δυο διαφορετικά συστήματα τα οποία εστιάζουν σε δυο ξεχωριστές προσεγγίσεις. Στη πρώτη, οι διαδρομές χαρτογραφούνται πριν από οποιαδήποτε επερώτηση ενώ στη δεύτερη τα μονοπάτια δημιουργούνται κατά τη διάρκεια των επερωτήσεων. Παρόλο που θεωρήσαμε τη δεύτερη προσέγγιση πιο κοντά στις απαιτήσεις που έχουμε θέσει για το σύστημα μας, αποφασίσαμε να κρατήσουμε και τις δυο υλοποιήσεις για λόγους συγκρίσεων.

2. Περιγραφή Συστήματος

Στην παρούσα πτυχιακή εργασία προσπαθήσαμε να δώσουμε λύσεις σε δυο βασικά προβλήματα στη διαχείριση των γράφων. Το πρώτο, είναι ο τρόπος με τον οποίο μπορούμε να διασχίσουμε ένα γράφο ενώ το δεύτερο είναι, πως μπορούμε να εκμεταλλευτούμε τα χαρακτηριστικά του γράφου (τις ιδιότητες των κόμβων και των ακμών του) προς όφελός μας.

Το πλήθος των δομικών στοιχείων ενός γράφου, δηλαδή των κόμβων και των ακμών του, καθορίζουν το πλήθος, την πολυπλοκότητα και το μήκος των μονοπατιών που μπορεί κάποιος να χρησιμοποιήσει για τη διάσχιση ενός γράφου από ένα κόμβο αφετηρία προς έναν άλλο τερματισμού. Σε κάθε επερώτηση (Query) λοιπόν σε μια βάση δεδομένων με γράφους, μια τέτοια διάσχιση λαμβάνει χώρα.

Εξαιτίας αυτών των χαρακτηριστικών, γίνεται αντιληπτό ότι, απαιτείται κόστος είτε σε χρόνο είτε σε χώρο είτε και στα δυο, προκειμένου να γίνει μια πλήρη καταγραφή των μονοπατιών που μπορούμε να διασχίσουμε.

Στα πλαίσια της πτυχιακής, αναπτύξαμε ένα σύστημα στο οποίο ο χρήστης μπορεί να θέτει κανόνες εφαρμογής ή τελεστές που θα χρησιμοποιηθούν στα επερωτήματα. Στο σύστημα μπορούμε να εισάγουμε τόσο άκυκλους κατευθυνόμενους γράφους (acyclic directed graphs) όσο και τυχαίους κατευθυνόμενους (arbitrary directed graphs).

Βασική λειτουργία του συστήματος είναι η ανεύρεση μονοπατιών όπου εφαρμόζοντας τους τελεστές στις ιδιότητες των δομικών στοιχείων του γράφου (πχ βάρη ακμών) να δημιουργούν νέα μονοπάτια από συνενώσεις ή συναθροίσεις των ήδη ανακτηθέντων. Τα χαρακτηριστικά των νέων μονοπατιών προκύπτουν από τις ιδιότητες των δομικών στοιχείων που μετέχουν στους υπολογισμούς.

Το μοντέλο της βάσης δεδομένων γράφων που χρησιμοποιείται στη παρούσα πτυχιακή είναι τύπου γράφου ιδιοτήτων (property graph). Αυτό θεωρείται μια από από τις πιο δημοφιλής μορφές του μοντέλου γράφων. Ένας γράφος ιδιοτήτων έχει τα εξής χαρακτηριστικά:

- Περιέχει κόμβους και ακμές
- Οι κόμβοι κατέχουν ιδιότητες (ζεύγη πεδίων-τιμών)
- Οι ακμές ονοματίζονται, είναι κατευθυνόμενες και διαθέτουν πάντα αρχή και

τέλος.

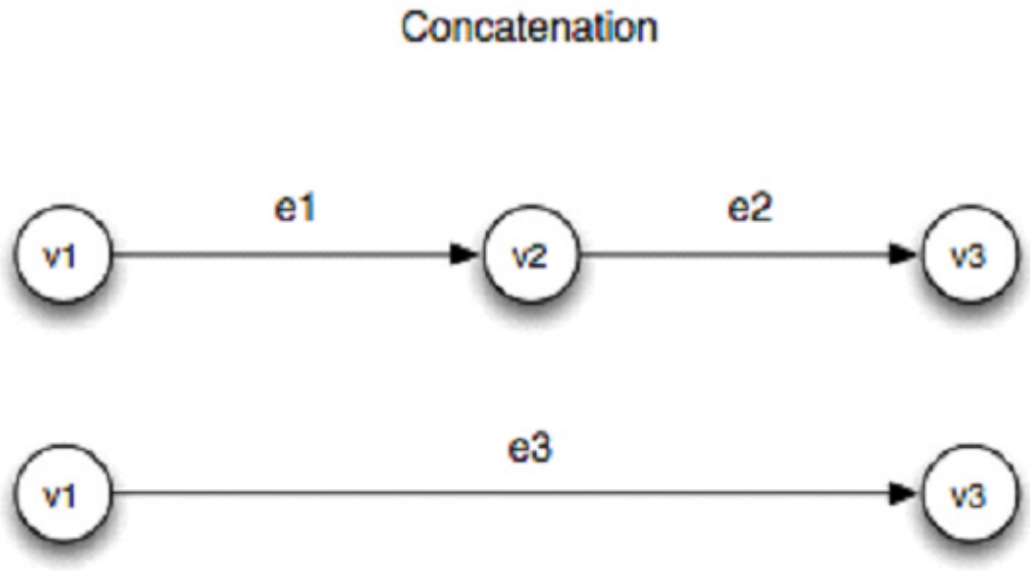
- Οι ακμές επίσης περιέχουν ιδιότητες

Ο περισσότερος κόσμος θεωρεί ότι το μοντέλο γράφων ιδιοτήτων είναι πιο διαισθητικό και εύκολο για να κατανοηθεί. Παρόλο απλό, μπορεί να χρησιμοποιηθεί για την περιγραφή της πλειονότητας των γράφων με τρόπους τέτοιους οι οποίοι παρέχουν χρήσιμες πληροφορίες για τα δεδομένα τους.

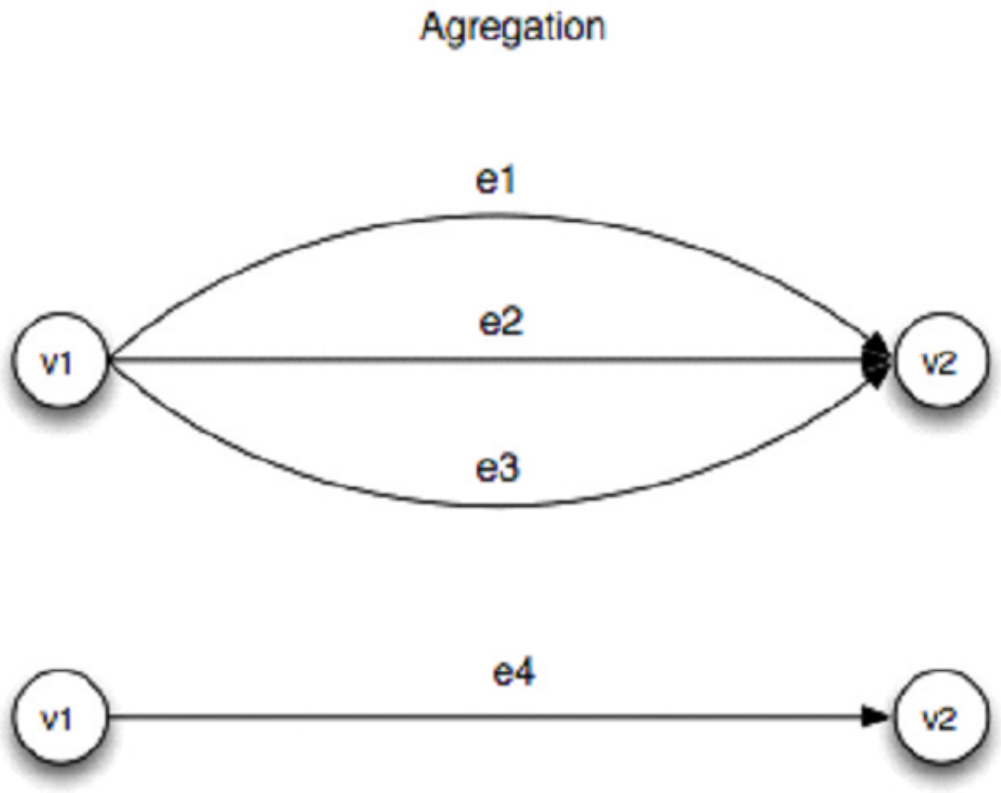
Μία επερώτηση αναζήτησης (search query) στα πλαίσια της πτυχιακής αυτής, είναι ο καθορισμός μιας ακμής από ένα κόμβο προς έναν άλλο. Οι κόμβοι αυτοί καλούνται αρχή (source) και τέλος (target) της επερώτησης. Η ακμή που τους συνδέει προκύπτει από τη σύνθεση ενός ή περισσότερων απλών μονοπατιών περιέχοντας έτσι και άλλους κόμβους διασυνδεδεμένους με τη σειρά τους με περισσότερες ακμές. Όπως έχει προαναφερθεί, στις πιο πολλές επερωτήσεις ο στόχος είναι να καθοριστούν αυτά τα μονοπάτια ακόμα και αν δεν μπορεί να σχηματιστεί μια μοναδική ακμή σαν αποτέλεσμα της επερώτησης.

Στα συστήματα που υλοποιήσαμε, ένας χρήστης μπορεί να ορίζει τελεστές ή κανόνες εφαρμογής. Οι κανόνες αυτοί είναι δύο ειδών, κανόνες συνένωσης (Concatenations) και κανόνες συνάθροισης (Aggregations). Χρησιμοποιούνται για τη σύνθεση νέων ακμών από τα δομικά στοιχεία των μονοπατιών μεταξύ των κόμβων των επερωτήσεων. Οι κανόνες εφαρμογής καθορίζουν ποιες ακμές θα μετέχουν στη δημιουργία των νέων ακμών βάση των ιδιοτήτων τους (πχ τύπους ακμών, τύπους κόμβων κλπ).

Με τον κανόνα συνένωσης μια νέα ακμή συντίθεται από την συνένωση δυο συνεχόμενων ακμών, όπως φαίνεται στο Σχήμα 1. Από την άλλη, ο κανόνας συνάθροισης συνθέτει μια νέα ακμή η οποία αντικαθιστά όλες εκείνες οι οποίες έχουν κοινή αρχή, κοινό τέλος και ίδιο τύπο (ιδιότητα “τύπος ακμής”), όπως φαίνεται στο Σχήμα 2.



Σχήμα 1: Παράδειγμα εφαρμογής κανόνα συνένωσης



Σχήμα 2: Παράδειγμα εφαρμογής κανόνα συνάθροισης

Σε κάθε γράφο όταν προσπαθούμε να τον διατρέξουμε μεταξύ δυο σημείων, το πρώτο πρόβλημα που καλούμαστε να αντιμετωπίσουμε είναι ποια διαδρομή να επιλέξουμε. Βεβαίως, μπορεί να μην υπάρχει μια μόνο διαδρομή αλλά πολύ περισσότερες αλλά όλες να έχουν κάποιο νόημα για μας.

Το πρόβλημα αυτό επιλύεται με το μεταβατικό εγκλεισμό (transitive closure) το οποίο μας επιδεικνύει αν δυο κόμβοι επικοινωνούν μεταξύ τους, ακόμα και όχι άμεσα συνδεδεμένοι. Λειτουργεί για ένα βάθος αναζήτησης d, δηλαδή αν ο κόμβος τέλους βρίσκεται d+1 σχέσεις μακρύτερα από τον κόμβο αρχής τότε λέμε ότι οι δυο κόμβοι δεν έχουν προσβασιμότητα (reachability).

Ο μεταβατικός εγκλεισμός δεν λύνει το πρόβλημα της επιλογής της διαδρομής. Δεν μας υπαγορεύει ποιους κόμβους ή ακμές θα πρέπει να διασχίσουμε για να φτάσουμε στον προορισμό μας. Μας είναι όμως χρήσιμο γιατί από τον κόμβο που βρισκόμαστε ή από τον κάποιο γειτονικό αναφέρει αν ο τελικός κόμβος είναι προσβάσιμος.

Το δεύτερο πρόβλημα είναι ο τρόπος που μπορούμε να εκμεταλλευτούμε τις πληροφορίες των μονοπατιών και πως να επιδεικνύουμε το αποτέλεσμα τους. Για την επίλυση του προβλήματος αυτού καταφύγαμε στη χρήση των τελεστών συνάθροισης και συνένωσης.

Για κάθε μονοπάτι που ξεκινάει από τον κόμβο αρχής με κατεύθυνση τον κόμβο τερματισμού, ελέγχουμε αν οι ακμές του πληρούν κατάλληλα κριτήρια συνένωσης ή συνάθροισης. Η εκτέλεση των υπολογισμών γίνεται με βάση τους κανόνες εφαρμογής, ενώ οι ακμές λαμβάνονται από τα μονοπάτια αρχίζοντας από το τέλος προς την αρχή. Αυτό ισχύει μιας και η αριστερή προσεταιριστικότητα του κανόνα εφαρμογής υποθέτει ότι κάθε προηγούμενος υπολογισμός έχει γίνει σε προηγούμενο βήμα εκτός και αν περιέχει την ακμή που προστέθηκε τελευταία.

Στο τέλος, ιδανικό αποτέλεσμα θα είναι εκείνο όπου θα έχει μεταφερθεί όλο το τμήμα του γράφου που διασχίζεται από ένα πλήθος μονοπατιών σε μια και μόνο ακμή που θα συνδέει τους τελικούς κόμβους. Η ακμή αυτή θα φέρει και όλες τις πληροφορίες που χρειαζόμαστε για το συγκεκριμένο επερώτημα. Η ιδανική κατάσταση δεν είναι πάντα εφικτή αλλά εξαρτάται τόσο από τον ίδιο το γράφο όσο και από τη γραμματική των κανόνων εφαρμογής.

Για την επίτευξη των ανωτέρω, η εκτέλεση των υπολογισμών θα λαμβάνει χώρα είτε σε κάθε βήμα είτε ανά πιο μεγάλα διαστήματα συγχρονισμού. Επειδή γνω-

ρίζουμε ότι οι ακμές διαφοροποιούνται μεταξύ τους αναλόγως της σχέσης που αντιπροσωπεύουν, μπορούμε να δημιουργήσουμε κανόνες εφαρμογής, ξεχωριστούς όχι μόνο για το είδος της ένωσης που θέλουμε να επιτύχουμε (συνένωση ή συνάθροιση ακμών) αλλά και αναλόγως της σχέσης της ακμής με τους κόμβους που συνδέει.

Αφήνοντας λοιπόν τον χρήστη να ορίσει τους κανόνες αυτούς, μπορεί να εκτελεί ποικίλα επερωτήματα λαμβάνοντας διαφορετικά αποτελέσματα αναλόγως των κανόνων που εφαρμόζονται κάθε φορά.

Όπως προαναφέρθηκε, υπάρχουν δυο είδη κανόνων εφαρμογής. Οι κανόνες εφαρμόζονται ξεχωριστά και έτσι μια ακμή η οποία προήλθε από την εφαρμογή του πρώτου μπορεί να μετέχει εν συνεχεία σε εφαρμογή του δεύτερου.

Ανά δυο ακμές του ενός μονοπατιού που πληρούν τα κριτήρια του πρώτου κανόνα, μετά τους υπολογισμούς προκύπτει μια νέα ακμή με αρχή τον κόμβο αρχής της πρώτης και τέλος τον κόμβο τέλους της δεύτερης. Η νέα ακμή τοποθετείται ξανά τελευταία στο ίδιο μονοπάτι, ενώ οι δύο ακμές που τη δημιούργησαν αφαιρούνται. Η διαδικασία αυτή επαναλαμβάνεται όσο υπάρχουν ακμές οι οποίες πληρούν τα κριτήρια.

Αντίθετα από ότι συμβαίνει παραπάνω, για την εφαρμογή του δεύτερου κανόνα απαιτούνται δυο στάδια. Στο πρώτο, συναθροίζονται όλες οι ακμές που πληρούν τα κριτήρια, η τελευταία από κάθε μονοπάτι, σε μια δομή προκειμένου να επεξεργαστούν στη συνέχεια.

Στο δεύτερο στάδιο εφαρμόζεται ο κανόνας σε όλες τις ακμές που συγκεντρώθηκαν παράγοντας μια νέα ακμή. Επειδή όλες οι ακμές που συγκεντρώθηκαν έχουν ίδια αρχή και ίδιο τέλος τότε και η νέα αρχή θα έχει τα ίδια χαρακτηριστικά. Η νέα ακμή τοποθετείται τελευταία σε όλα τα μονοπάτια από όπου ελήφθησαν οι ακμές που τη δημιούργησαν.

Ένα τέτοιο σύστημα διαχείρισης γράφων μπορεί να έχει πρακτική χρησιμότητα όπως να δίνει απαντήσεις σε ερωτήματα συνδεσιμότητας, πιθανοτήτων αρέσκειας κλπ. Όπως είπαμε και στην αρχή δημιουργήθηκαν δυο ξεχωριστές υλοποιήσεις του συστήματος αυτού προκειμένου να συμπεριλάβουμε τις δυο διαφορετικές θεωρητικές προσεγγίσεις

Και οι δυο υλοποιήσεις ανταλλάσσουν πληροφορίες μέσω των διεργασιών αιτήματος. Κοινό στοιχείο των υλοποιήσεων είναι ένα είδος διεργασιών οι οποίες

περιέχουν τα μονοπάτια που έχουν προσπελάσει. Αυτές οι διεργασίες περιέχουν τις ακμές των μονοπατιών και χρησιμοποιούν σαν στοιχεία αναφοράς τους κόμβους, δηλαδή κατά την εκτέλεση της διεργασίας ελέγχεται ο κόμβος τον οποίο αφορά η εκάστοτε διεργασία και συμπεριλαμβάνονται είτε οι ακμές είτε ο ίδιος ο κόμβος στα μονοπάτια προσπέλασης.

Όταν ο γράφος φορτώνεται στο σύστημα τότε μια σειρά από δομές και αντικείμενα αναλαμβάνουν να παίξουν το ρόλο των δομικών του στοιχείων. Έτσι, οι κόμβοι περιγράφονται με αντικείμενα τα οποία διατηρούν τα χαρακτηριστικά των κόμβων του γράφου όπως τύπο, όνομα και λοιπές ιδιότητες, αλλά διατηρούν επίσης και κάποιες δομές όπως λίστες με τις εισερχόμενες προς τον κόμβο ή εξερχόμενες προς τον κόμβο ακμές.

Αντίστοιχα, το αντικείμενο που περιγράφει μια ακμή περιέχει πληροφορίες όπως τον κόμβο αρχής, τον κόμβο τέλους, τον τύπο της και οποιαδήποτε άλλη ιδιότητα καθορίζεται από τον εκάστοτε γράφο.

Από τα παραπάνω γίνεται αντιληπτό ότι δοθέντος ενός κόμβου, μπορούμε να χρησιμοποιήσουμε τις εξερχόμενες ακμές του και να μεταπηδήσουμε στον επόμενο γειτονικό κόμβο (σε βάθος δηλαδή d = 1).

Στο σύστημα δυο σταδίων, το σύστημα καλείται έτσι γιατί αποτελείται από δυο στάδια. Στο πρώτο, γίνεται ένα σχεδιάγραμμα δηλαδή κατασκευάζουμε το γράφο εξαρτήσεων, των διαδρομών από τον κόμβο αρχής μέχρι τον κόμβο τέλους. Εκεί, σε ξεχωριστές δομές καταγράφονται για τον κάθε κόμβο από ποιους κόμβους πρέπει να περιμένει μηνύματα και σε ποιους κόμβους πρέπει εκείνος να στείλει.

Στο δεύτερο στάδιο, δημιουργείται μια αλυσίδα μηνυμάτων τα οποία περιλαμβάνουν τους κόμβους του γράφου μέχρις ότου τα μηνύματα να περιέχουν τον τελικό κόμβο της αναζήτησης.

Δεδομένου ότι έχει ήδη πραγματοποιηθεί η χαρτογράφηση των διαδρομών, κάθε κόμβος είναι ενήμερος από ποιους κόμβους περιμένει να λάβει και προς ποιους κόμβους θα στείλει με τη σειρά του. Τα μηνύματα προς τους επόμενους κόμβους αποστέλλονται μετά τη λήψη όλων των μηνυμάτων από εκείνους που αναμένει. Τότε είναι που εκτελούνται και οι υπολογισμοί των κανόνων εφαρμογής και έτσι τα μηνύματα προς τους επόμενους κόμβους περιλαμβάνουν ήδη τις νέες ακμές που έχουν δημιουργηθεί από την εκτέλεση των παραπάνω υπολογισμών. Για το λόγο αυτό λέμε ότι κάθε κόμβος είναι δυνητικά τόπος εκτέλεσης υπολογισμών

κανόνων εφαρμογής.

Στο σύστημα ενός σταδίου, το στάδιο της χαρτογράφησης παραλείπεται. Από την αρχή, δημιουργούνται ξεχωριστά ερωτήματα για κάθε επερώτηση. Όπως και προηγουμένως, κάθε διεργασία περιέχει τον κόμβο παραλήπτη και τη διαδρομή που διέρχεται από τον κόμβο αυτό. Μια σημαντική διαφορά από την αρχική υλοποίηση είναι ότι για κάθε ξεχωριστή διαδρομή υπάρχει και μια ξεχωριστή διεργασία αναζήτησης.

Στο σύστημα αυτό δεν επιχειρούμε να κατασκευάσουμε το γράφο εξαρτήσεων, αλλά εκτελούμε τους υπολογισμούς σε κάθε εκτέλεση αναζήτησης διαδρομής και σε σημεία συγχρονισμού. Όταν εντοπίζονται τυχόν λάθη τότε εκτελούνται ειδικές ενέργειες για διόρθωση των υπολογισμών.

Μια ακόμη ειδοποιός διαφορά από την προηγούμενη υλοποίηση, είναι το γεγονός ότι κάθε διαδρομή είναι αυτόνομη μέσα στο σύστημα και οι υπολογισμοί δεν γίνονται στα πλαίσια κάποιου κόμβου. Αντιθέτως, κάθε διεργασία αιτήματος διαδρομής αφορά μια ανεξάρτητη διαδρομή και κατά την εκτέλεσή της,

- εκτελούνται άμεσα οι υπολογισμοί των συνενώσεων και κατόπιν,
- ελέγχεται αν η τελική της ακμή της διαδρομής πληροί τους κανόνες συνάθροισης και αν ναι, τότε αυτή αποθηκεύεται σε μια κατάλληλη δομή.
- αν η ακμή δεν πληρεί τα κριτήρια των κανόνων συνάθροισης, τότε η διαδρομή κατασκευάζει νέα μονοπάτια από τον κόμβο τέλους προς τους γειτονικούς κόμβους που συμφωνούν με το μεταβατικό εγκλεισμό. Για κάθε ένα μονοπάτι που δημιουργείται, παράγεται μια νέα διεργασία αναζήτησης διαδρομής και τοποθετείται στην ουρά εκτέλεσης.

3. Συμπεράσματα

Σε γενικές γραμμές για το μεγαλύτερο μέρος των σεναρίων που δοκιμαστήκαν, τα δυο συστήματα έχουν παρεμφερή συμπεριφορά όσον αφορά το συνολικό χρόνο εκτέλεσης μιας επερώτησης. Οι διαφορές τους εντοπίζονται στο ότι κατά την πρώτη υλοποίηση το μεγάλο μέρος του συνολικού χρόνου εκτέλεσης αποτελεί η δημιουργία του σχήματος εξαρτήσεων, ενώ το χρονικό κόστος των

υπολογισμών πάνω στα μονοπάτια που δημιουργούνται είναι συγκριτικά χαμηλό. Επιπλέον το σύστημα αυτό είναι βεβαρημένο χωρικά καθώς διατηρεί στη μνήμη το σύνολο των ακμών που συντελούν τα μονοπάτια τα οποία μετέχουν στην επερώτηση.

Η δεύτερη υλοποίηση ανάγει το συνολικό χρόνο εκτέλεσης της επερώτησης σε κόστος υπολογισμών, χωρίς να επιβαρύνεται επιπλέον σε κόστος μνήμης.

Από τα διάφορα σενάρια με τα οποία εξετάσθηκε η ευαισθησία εισόδου των συστημάτων, δεν διαφαίνεται με σαφήνεια η υπεροχή του ενός έναντι του άλλου, καθώς και τα δυο παρουσιάζουν θετικές και αρνητικές πτυχές.

Τέλος και οι δύο υλοποιήσεις, δεδομένου ότι είναι εφικτή η εφαρμογή των κανόνων συνένωσης και συνάθροισης φράσσονται από τετραγωνική πολυπλοκότητα. Το γεγονός αυτό καθιστά τις ιδέες που παρουσιάζονται μέσω αυτών των συστημάτων είναι ιδιαίτερα ανταγωνιστικές σε σχέση με τα υπάρχοντα NoSQL συστήματα.

Αναφορές

[1] Yannis Ioannidis, Maria Vayanou, Thodoris Georgiou, Katerina Iatropoulou, Manos Karvounis, Vivi Katifori, Marialena Kyriakidi, Natalia Manola, Alexandros Mouzakidis, Lefteris Stamatogiannakis, Mei Li Triantafyllidi, “Profiling Attitudes for Personalized Information Provision”, MaDgIK Lab, Dept. of Informatics & Telecom, Univ. of Athens, IEEE, 2011

[2] Mike Buerli, Current State of Graph Databases, Cal Poly San Luis Obispo, 2012

[3] I. Robinson, J. Webber and E. Eifrem, Graph Databases, Oreilly, 2013

[4] Knuth, Donald E., The Art Of Computer Programming Vol 1. 3rd ed., Boston: Addison-Wesley, 1997

[5] GraphML Project Group, “GraphML Specification”, http://graphml.graphdrawing.org/specification.html [Προσπελάστηκε 30/05/14]

[6] The Stony Brook Algorithm Repository, “Transitive Closure and Reduction”, http://www.cs.sunysb.edu/~algorith/files/transitive-closure.shtml

[Προσπελάστηκε 30/05/14]

[7] Network Working Group, “A Universally Unique IDentifier (UUID) URN Namespace”, http://tools.ietf.org/html/rfc4122 [Προσπελάστηκε 30/05/14]

Κατανεμημένος και Δυναμικός Χρονοπρογραμματισμός

Ευάγγελος Ν. Νικολόπουλος (vgnikolop@di.uoa.gr)
Ιωάννης Φ. Χρόνης (i.chronis@di.uoa.gr)

Περίληψη

Λόγω του μεγάλου όγκου των δεδομένων που απαιτείται να επεξεργαστουν από τα σύγχρονα υπολογιστικα συστήματα χρήση κατανεμημένων συστημάτων είναι απαραίτητη. Η πιο διαδεδομένη οργάνωση των κατανεμημένων συστημάτων είναι το master-worker μοντέλο. Ένας κόμβος αναλαμβάνει την ανάθεση, την παρακολούθηση και τον συγχρονισμό των εργασιών στους υπόλοιπους κόμβους βάση ενός προαποφασισμένου σχεδίου, του χρονοπρογράμματος. Τα αδύναμα σημεία αυτής της προσέγγισης είναι ότι δημιουργείται ένα κεντρικό σημείο διαχείρισης που καθώς αυξάνεται το πλήθος των μηχανημάτων ενός κατανεμημένου συστήματος μπορεί να αποτελέσει σημείο συμφόρησης. Επίσης καθώς το χρονοπρόγραμμα αποφασίζεται πριν την εκτέλεση, η δημιουργία του βασίζεται σε εκτιμήσεις και στατιστικά που πολλές φορές δεν είναι ικανά να παράξουν το βέλτιστο αποτέλεσμα. Στην εργασία αυτή προσπαθούμε να αντιμετωπίσουμε τα δύο παραπάνω προβλήματα, το καθένα ξεχωριστά. Στη πρώτη περίπτωση μεταφέραμε μέρος της λειτουργικότητας του κεντρικού κόμβου διαχείρισης σε κάθε κόμβο και δεν δημιουργείται σημείο συμφόρησης. Στη δεύτερη περίπτωση μεταβάλλουμε δυναμικά το χρονοπρόγραμμα κατά τη διάρκεια της εκτέλεσής του με βάση μετρικές που ορίζουμε.

Λέξεις κλειδιά: Δυναμικός Χρονοπρογραμματισμός, Κατανεμημένος Χρονοπρογραμματισμός, Κατανεμημένα Συστήματα, Υπολογιστικά Νέφη.

Επιβλέποντες

Ιωάννης Ιωαννίδης, Καθηγητής, Herald Kllapi, Υποψήφιος Διδάκτωρ.

1. Εισαγωγή

Ένας βέλτιστος χρονοπρογραμματιστής επιτυγχάνει μέγιστη χρήση όλων των πόρων, αποφυγή δημιουργίας σημείων συμφόρησης και ελαχιστοποίηση του χρόνου αδράνειας των μηχανημάτων. Η διαδικασία αυτή είναι αρκετά δύσκολη καθώς πρέπει να ληφθούν υπόψιν πολλές παράμετροι όπως η μορφολογία των δεδομένων, του δικτύου και να υπολογιστεί ο αναμενόμενος χρόνος εκτέλεσης κάθε εργασίας.

Ο χρονοπρογραμματιστής μπορεί να λειτουργεί είτε έχοντας λάβει όλες τις αποφάσεις εξ' αρχής είτε δυναμικά [3,4] κατά τη διάρκεια της εκτέλεσης. Ως προς την επίβλεψη και την διαχείριση της εκτέλεσης μπορεί ή ένας κεντρικός κόμβος να είναι υπεύθυνος για τον συγχρονισμό και τον διαμοιρασμό των εργασιών ή κάθε μηχάνημα να είναι υπεύθυνο για την εκτέλεση των δικών του εργασιών και την ενημέρωση των υπολοίπων σε ότι αφορά τις δικές του εργασίες.

Στο τμήμα μας αναπτύσσεται το Exareme. Το Exareme [1] είναι ένα κατανεμημένο σύστημα επεξεργασίας ροών δεδομένων σε συστάδες υπολογιστών (clusters) ή υπολογιστικά νέφη (clouds). Το σύστημα προσφέρει μια δηλωτική γλώσσα βασισμένη στην SQL η οποία μπορεί να εκφράσει πολύπλοκες ροές που επεξεργάζονται μεγάλου όγκου δεδομένα. Το Exareme αρχικά βασιζόταν στο στατικό κεντρικοποιημένο μοντέλο. Αυτή η εργασία χωρίζεται σε δυο μέρη. Πρώτον ο μετασχηματισμός του υποσυστήματος χρονοπρογραμματισμού σε στατικό κατανεμημένο και δεύτερον υλοποίηση ενός υποσυστήματος δυναμικού κεντρικοποιημένου χρονοπρογραμματισμού.

2. Παρουσίαση Συστήματος

Από τους πρωταρχικούς στόχους για την ανάπτυξη του συστήματος, ήταν η επεξεργασία όσο το δυνατό μεγαλύτερου όγκου δεδομένων με όσο το δυνατό μικρότερο κόστος. Το σύστημά μας, λοιπόν, χτίστηκε πάνω από ένα στατικό σύστημα βάσεων δεδομένων, το MadIS [2]. Το Exareme αποτελεί ένα σύστημα αποθήκευσης και επεξεργασίας μεγάλου όγκου δεδομένων χρησιμοποιώντας συστάδες υπολογιστών (clusters).

Η γλώσσα ερωτημάτων του Exareme είναι η AQL, μια υψηλού επιπέδου γλώσσα για

επεξεργασία γενικών ερωτημάτων. Ένα ερώτημα AQL μεταφράζεται αρχικά σε μια χαμηλότερου επιπέδου γλώσσα, την ADFL, που είναι μια γλώσσα για διαχείριση ροών δεδομένων. Για την αποτίμηση ενός AQL ερωτήματος, αρχικά το ερώτημα μεταφράζεται σε ADFL. Το query processor επεξεργάζεται το ερώτημα και μαζί με τον χρονοπρογραμματιστή παράγει το πλάνο εκτέλεσης. Τέλος, το πλάνο παραδίδεται στη μηχανή εκτέλεσης που είναι υπεύθυνη για την υλοποίησή του.

Η πτυχιακή αυτή θα εστιάσει και θα κάνει αλλαγές στο σύστημα από το στάδιο της δημιουργίας του πλάνου και μετά. Τα βασικά υποσυστήματα που θα εξετάσουμε είναι το πλάνο εκτέλεσης, η μηχανή εκτέλεσης, ο υποδοχέας (container) και ο μηχανισμός μεταφοράς δεδομένων.

2.1. Πλάνο Εκτέλεσης

Το πλάνο εκτέλεσης είναι οι οδηγίες που παράγονται από το optimization engine του Exareme και υλοποιείται από το runtime κομμάτι του συστήματος. Το πλάνο περιέχει δηλώσεις για τους υποδοχείς, τους τελεστές, τις ενδιάμεσες μνήμες και τις διασυνδέσεις τους. Το πλάνο αναπαρίσταται από ένα γράφο, οι κόμβοι του είναι τελεστές και οι ακμές του ροές δεδομένων (εξαρτήσεις). Κατά τη δημιουργία του πλάνου εκτέλεσης το optimization engine λαμβάνει υπόψιν του περιορισμούς που ορίζονται από την κατανομή των δεδομένων στους υποδοχείς, από τις εξαρτήσεις τελεστών από τα αποτελέσματα άλλων τελεστών και από εκτιμήσεις του χρόνου εκτέλεσης των τελεστών που έχει δημιουργήσει από τα στατιστικά προηγούμενων εκτελέσεων και από τη γνώση των δεδομένων.

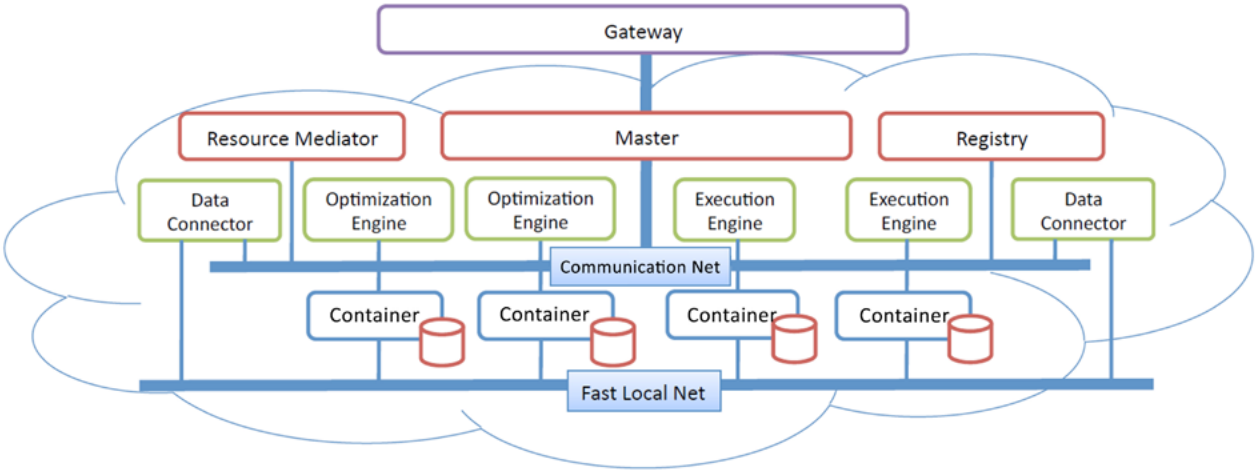
2.2. Μηχανή Εκτέλεσης (Execution Engine)

Η μηχανή εκτέλεσης (Execution Engine) είναι υπεύθυνη για την εκτέλεση και τον συντονισμό των τελεστών (operators), των ενδιάμεσων μνημών (buffers) και γενικότερα των συνόδων (sessions). "Όταν ένας τελεστής είναι έτοιμος να τρέξει, η μηχανή εκτέλεσης τον αναθέτει στον αντίστοιχο container, παρακολουθεί τη λειτουργία του και ενημερώνεται για το τερματισμό του. Όταν τερματίζει ένας τελεστής η μηχανή εκτέλεσης με βάση το πλάνο επικοινωνεί με τους αντίστοιχους υποδοχείς και προγραμματίζει τους επόμενους τελεστές.

2.3. Υποδοχείς (Containers)

Οι υποδοχείς δεν έχουν έλεγχο πάνω στον προγραμματισμό και την εκτέλεση του πλάνου. Είναι υπεύθυνοι για τη δημιουργία των τελεστών, της ενδιάμεσης μνήμης, τον διαχειριστή των προσαρμογέων και του διαχειριστή των τελεστών όταν λάβουν τα κατάλληλα μηνύματα από την κεντρική μηχανή εκτέλεσης.

2.4. Αρχιτεκτονική



Σχήμα 1: Αρχιτεκτονική του συστήματος

Η συνολική αρχιτεκτονική του συστήματος φαίνεται στο σχήμα 1. Τα ερωτήματα που στέλνονται από τους Clients, περνούν δια μέσου της μονάδας Gateaway στον κόμβο Master. Το σύστημα αποτελείται από ένα σύνολο από Optimization Engines, Execution Engines και Workers. Ο Master δίνει αρχικά το ερώτημα σε μία από τις Optimization Engines.

Οι Optimization Engines παράγουν ένα πλάνο εκτέλεσης. Το πλάνο εκτέλεσης που παράγεται δίνεται σε μια Execution Engine. Η Execution Engine επικοινωνεί με τον Resource Manager, ώστε να δεσμευτούν οι πόροι που είναι απαραίτητοι για την εκτέλεση του ερωτήματος. Μετά από την δέσμευση ξεκινά η δρομολόγηση των operators στους Workers, με σεβασμό στις εξαρτήσεις μεταξύ τους και τους διαθέσιμους πόρους. Είναι επίσης υπεύθυνος για την επανεκτέλεση των τελεστών εκείνων που οι εκτέλεση τους απέτυχε. Τέλος η επικοινωνία όλων τον μονάδων του συστήματος γίνεται μέσω καλά ορισμένων πρωτοκόλλων επικοινωνίας που

έχει οριστεί πάνω από το τοπικό δίκτυο.

2.5. Ανάλυση Μηνυμάτων (ή επικοινωνίας)

Ας μελετήσουμε ένα re-partition ερώτημα, με τον αρχικό πίνακα να βρίσκεται σε n διαμερίσεις και θα μοιραστεί σε n νέες διαμερίσεις. Θα χρειαστούν: n select τελεστές, n τελεστές αποθήκευσης, 2n ενδιάμεσες μνήμες και $2n^2$ links.

Τα αντίστοιχα μηνύματα που χρειάζονται είναι: 2n για τη δημιουργία των τελεστών, 2n για τη δημιουργία των ενδιάμεσων μνημών, $2n^2$ για τη δημιουργία των links και 2n για τον τερματισμό των τελεστών.

Τελικά απαιτούνται: $6n+2n^2$ μηνύματα για ένα re-partition ερώτημα από n σε n διαμερίσεις.

3. Κατανεμημένος Χρονοπρογραμματισμός

Στο κεφάλαιο αυτό θα παρουσιάσουμε τις αλλαγές που κάναμε στο σύστημα για να μετατρέψουμε το υποσύστημα του χρονοπρογραμματισμού από κεντρικοποιημένο σε κατανεμημένο. Επίσης θα μελετήσουμε τις επιπτώσεις των αλλαγών στην απόδοση του συστήματος.

3.1. Πρόταση

Η βασικότερη αλλαγή που κάναμε ήταν να μεταφέρουμε μεγάλο μέρος της λειτουργικότητας της μηχανής εκτέλεσης στους υποδοχείς. Για να πετύχουμε αυτό, προσθέσαμε μια τοπική μηχανή εκτέλεσης σε κάθε υποδοχέα.

Ο ρόλος της κεντρικής μηχανής εκτέλεσης περιορίζεται στην δημιουργία, τον διαμοιρασμό του πλάνου, την δημιουργία καναλιών για την απευθείας επικοινωνία των υποδοχέων, τον συγχρονισμό έναρξης λήξης εκτέλεσης του πλάνου και την συγκέντρωση των στατιστικών εκτέλεσης. Διαδικασίες που απαιτούν σταθερό υπολογιστικό κόστος ως προς το μέγεθος και τη πολυπλοκότητα του πλάνου και γραμμικό ως προς το πλήθος των υποδοχέων.

Η τροποποιημένη, τοπική μηχανή εκτέλεσης που προσθέσαμε εκτελεί από το

πλάνο που έχει λάβει το κομμάτι που αντιστοιχεί στον υποδοχέα που βρίσκεται και αναλαμβάνει τον συγχρονισμό με άλλους υποδοχείς, όπου είναι απαραίτητο. Προσθέσαμε μια ουρά σε κάθε υποδοχέα ώστε οι τελεστές που προγραμματίζονται από τη τοπική μηχανή εκτέλεσης να εκτελούνται μόνο όταν υπάρχουν διαθέσιμοι πόροι. Έτσι πλέον υπάρχει μια κεντρική μηχανή εκτέλεσης και από μία τοπική μηχανή εκτέλεσης σε κάθε container.

3.2. Ροή εκτέλεσης

Η κεντρική Μηχανή Εκτέλεσης λαμβάνει το πλάνο από το optimization engine, δημιουργεί τα κανάλια επικοινωνίας μεταξύ των τοπικών μηχανών εκτέλεσης και τους προωθεί το πλάνο. Ο αλγόριθμος που ακολουθεί η τοπική μηχανή εκτέλεσης είναι ο εξής:

- Βρίσκει από το πλάνο μέσω του dependency solver ποιοι από τους έτοιμους προς εκτέλεση τελεστές αντιστοιχούν στον τοπικό υποδοχέα και τους τοποθετεί στην ουρά εκτέλεσης.
- Ο υποδοχέας παρακολουθεί την ουρά εκτέλεσης και εκτελεί τους τελεστές που βρίσκονται εκεί όσο υπάρχουν διαθέσιμοι πόροι.
- Κάθε φορά που ένας τελεστής τελειώνει, ο υποδοχέας στον οποίο έτρεχε ειδοποιεί την τοπική μηχανή εκτέλεσης.
- Αυτή με τη σειρά της ειδοποιεί τις τοπικές μηχανές εκτέλεσης των άλλων υποδοχέων στους οποίους υπάρχει τελεστής που έχει εξάρτηση με τον πρώτο.
- Η τοπική μηχανή εκτέλεσης επαναλαμβάνει αυτή τη διαδικασία μέχρι να εκτελεστούν όλοι οι τελεστές του υποδοχέα όπου και ενημερώνει τη κεντρική μηχανή εκτέλεσης ότι ολοκλήρωσε το μέρος του πλάνου που της αντιστοιχούσε.

Τέλος η κεντρική μηχανή εκτέλεσης περιμένει να λάβει από όλες τις τοπικές το παραπάνω μήνυμα ώστε να ολοκληρωθεί το πλάνο.

3.3. Ανάλυση Μηνυμάτων (ή επικοινωνίας)

Ας μελετήσουμε ένα re-partition ερώτημα με m υποδοχείς, τον αρχικό πίνακα να βρίσκεται σε n διαμερίσεις ο οποίος μοιράζεται σε n νέες διαμερίσεις. Θα χρεια-

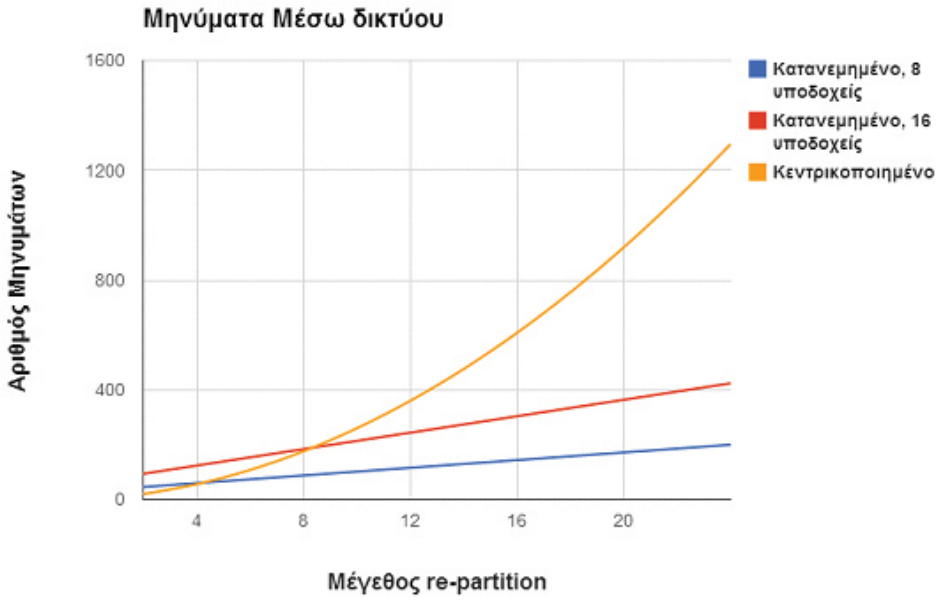
στούν: n select τελεστές, n τελεστές αποθήκευσης, 2n ενδιάμεσες μνήμες και $2n^2$ links.

Τα αντίστοιχα μηνύματα που χρειάζονται είναι: 2n για τη δημιουργία των τελεστών, 2n για τη δημιουργία των ενδιάμεσων μνημών, $2n^2$ για τη δημιουργία των links, 2n για τον τερματισμό των τελεστών και επιπλέον 4m μηνύματα για την έναρξη και τον τερματισμό των υποδοχέων και m(n-n/m) μηνύματα σε απομακρυσμένη μηχανή εκτέλεσης για την ενημέρωση τερματισμού ενός τελεστή.

Συνολικά απαιτούνται: $6n+2n^2+ 4m+ mn-n$ μηνύματα για ένα re-partition από n σε n διαμερίσεις. Σε αυτή την έκδοση τα μηνύματα είναι περισσότερα συνολικά αλλά μοιράζονται στις τοπικές μηχανές εκτέλεσης και έτσι αποφεύγεται η δημιουργία σημείου συμφόρησης στην μηχανή εκτέλεσης. Επίσης πολλά από τα μηνύματα διαχειρίζονται τοπικά και δεν επιβαρύνουν το δίκτυο.

Συγκεκριμένα, μέσω του δικτύου μεταφέρονται 4m+mn-n μηνύματα και από αυτά, τα 4m από και προς τη κεντρική μηχανή εκτέλεσης ενώ τα υπόλοιπα mn-n μεταξύ των υποδοχέων.

Παρατηρούμε ότι για μικρά πλάνα εκτέλεσης το κεντρικοποιημένο σύστημα απαιτεί λιγότερα μηνύματα και αποδίδει καλύτερα (Σχήμα 2). Όσο αυξάνεται το μέγεθος του πλάνου, το κατανεμημένο σύστημα απαιτεί όλο και λιγότερα μηνύματα σε σχέση με το κεντρικοποημένο.



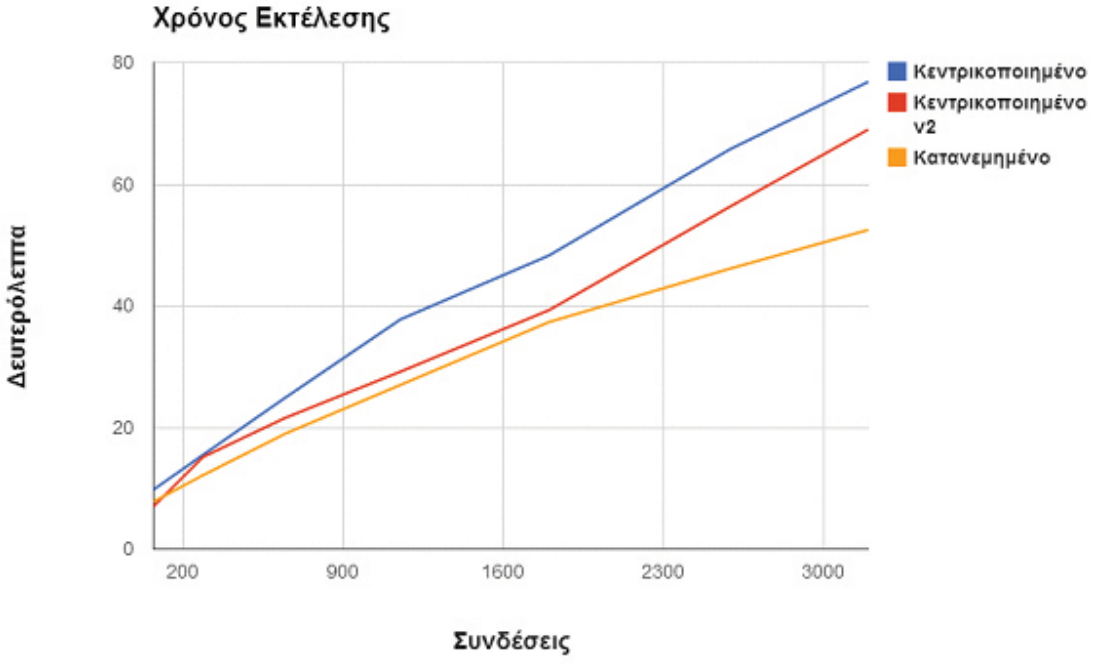
Σχήμα 2: Αριθμός μηνυμάτων σε σχέση με το μέγεθος re-partition

3.4. Πειραματική αξιολόγηση

Πειραματική Διάταξη. Οι δοκιμές έγιναν σε τρεις εικονικές μηχανές που φιλοξενήθηκαν στο okeanos του grnet με 1 cpu core (2Ghz),4 GB RAM και η ταχύτητα του δικτύου που τις ενώνει κυμαίνεται από 10 μέχρι 15mb/sec.

Για τις μετρήσεις μας τρέξαμε re-partition ερωτήματα με ίσο αρχικό και τελικό αριθμό διαμερίσεων. Τα ερωτήματα αυτά επιβαρύνουν περισσότερο το υποσύστημα χρονοπρογραμματισμού καθώς χρειάζονται μεγάλο αριθμό μηνυμάτων συγχρονισμού. Χρησιμοποιήσαμε ως πλήθη των διαμερίσεων 3, 6, 9, 12, 15, 18 και 20. Τρέξαμε την έκδοση με την αρχική έκδοση κεντρικοποιημένο χρονοπρογραμματιστή (Κεντρικοποιημένο), τη βελτιωμένη έκδοση με τον κεντρικοποιημένο χρονοπρογραμματιστή (Κεντρικοποιημένο v2) και την έκδοση με τον κατανεμημένο χρονοπρογραμματιστή (D) και μετρήσαμε τον συνολικό χρόνο εκτέλεσης.

Αποτελέσματα. (Σχήμα 3) Όπως περιμέναμε ο χρόνος εκτέλεσης μειώθηκε με τη χρήση του κατανεμημένου συστήματος χρονοπρογραμματισμού. Επίσης όσο αυξάνεται ο αριθμός των συνδέσεων παρατηρούμε οτι η διαφορά στο χρόνο αυξάνεται.



Σχήμα 3: Χρόνος εκτέλεσης σε σχέση με τον αριθμό συνδέσεων

3.5. Συμπέρασμα

Η καθυστέρηση στην επικοινωνία που παρατηρήθηκε στο σύστημα με τον κεντρικοποιημένο χρονοπρογραμματιστή και οφείλεται στη λειτουργία της κεντρικής μηχανής εκτέλεσης ως μοναδικού σημείου συγχρονισμού και ελέγχου ολόκληρου του συστήματος ήταν ο λόγος για τη δημιουργία του κατανεμημένου χρονοπρογραμματιστή.

Με τις αλλαγές που κάναμε δεν υπάρχει πια ένα κεντρικό σημείο ελέγχου. Αναθέσαμε τις λειτουργίες της κεντρικής μηχανής εκτέλεσης σε τροποποιημένες εκδόσεις της που τοποθετήσαμε σε κάθε υποδοχέα. Αυτό ήταν εφικτό καθώς το πλάνο εκτέλεσης προαποφασίζεται και άρα υπάρχει η δυνατότητα να μοιραστεί στους υποδοχείς πριν από την αρχή της εκτέλεσης. Καταφέραμε, επίσης, να μειώσουμε τα μηνύματα που στέλνονται μέσω του δικτύου και αυτά που χρησιμοποιούν το δίκτυο να μην έχουν ως προορισμό το ίδιο σημείο.

Από τις μετρήσεις φαίνεται η βελτίωση του χρόνου εκτέλεσης και πιστεύουμε ότι και σε ακόμη μεγαλύτερα ερωτήματα και μεγαλύτερα cluster η διαφορά θα αυξηθεί.

4. Δυναμικός Χρονοπρογραμματισμός

Η απόδοση ενός συστήματος ανάγεται στην σωστή αξιοποίηση των πόρων του. Στο δικό μας σύστημα αυτό σημαίνει αποδοτική κατανομή των τελεστών στους υποδοχείς. Για να δημιουργηθεί ένα αποδοτικό πλάνο χρειάζεται καλή γνώση των δεδομένων, ακριβή πρόβλεψη του πλήθους των ενδιάμεσων αποτελεσμάτων και του υπολογιστικού κόστους που απαιτείται. Οι εκτιμήσεις αυτές μπορούν να παραχθούν είτε από μοντέλα είτε από στατιστικά που έχουν συλλεχθεί από προηγούμενες εκτελέσεις. Είναι προφανές ότι αν οι εκτιμήσεις δεν είναι ακριβείς δεν μπορεί να παραχθεί το αποδοτικό πλάνο. Επιπλέον και άλλοι παράγοντες όπως η δυσλειτουργία ενός μικρού κομματιού του συστήματος (ένας υπολογιστικός ή δικτυακός κόμβος) μπορεί να προκαλέσει δυσανάλογα μεγάλη καθυστέρηση στη συνολική εκτέλεση του πλάνου.

Στο κεφάλαιο αυτό θα παρουσιάσουμε τις αλλαγές που κάναμε στο σύστημα για να μετατρέψουμε το υποσύστημα του χρονοπρογραμματισμού από κεντρικοποιημένο σε δυναμικό. Επίσης θα μελετήσουμε τις επιπτώσεις των αλλαγών στην

απόδοση του συστήματος.

4.1. Πρόταση

Ο δυναμικός χρονοπρογραμματισμός μεταφέρει τη διαδικασία της ανάθεσης μιας δουλειάς σε ένα εργάτη από το χρόνο δημιουργίας του πλάνου στην στιγμή λίγο πριν την εκτέλεση της δουλειάς. Στο σύστημα μας οι τελεστές αντιστοιχούνται σε υποδοχείς κατά τη διάρκεια της εκτέλεσης από τον χρονοπρογραμματιστή ο οποίος έχει πλήρη γνώση της κατάστασης του συστήματος. Με αυτό τον τρόπο μπορούν να αποφευχθούν καθυστερήσεις από λάθος προβλέψεις, αλλά και από συμφορήσεις στο δίκτυο ή από υπερβολικό φόρτο σε κάποιον υποδοχέα.

4.2. Ροή εκτέλεσης

Ο optimizer δημιουργεί το δυναμικό πλάνο. Σε αυτό το πλάνο μόνο οι τελεστές που επενεργούν σε υπάρχοντα δεδομένα υπάρχουν αντιστοιχισμένοι με υποδοχείς. Οι υπόλοιποι τελεστές, ενδιάμεσες μνήμες και σύνδεσμοι ομαδοποιούνται από τον optimizer έτσι ώστε εσωτερικά κάθε ομάδας να μην υπάρχουν εξαρτήσεις.

Η Μηχανή Εκτέλεσης λαμβάνει το πλάνο από το optimization engine. Ο αλγόριθμος που ακολουθεί είναι ο εξής,

- στέλνει στους υποδοχείς μηνύματα για να ξεκινήσουν τους ήδη αντιστοιχισμένους τελεστές
- κρατάει στατιστικά για το πλήθος των τελεστών που τρέχουν ή περιμένουν στην ουρά κάθε υποδοχέα
- μόλις λάβει μήνυμα τερματισμού ενός τελεστή:
  - μαζεύει στατιστικά (χρόνο εκτέλεσης, δεδομένα που παρήγαγε κτλ)
  - ενημερώνει τις δομές που παρακολουθούν την κατάσταση του συστήματος
  - ελέγχει αν υπάρχουν ομάδες έτοιμες προς αντιστοίχηση με υποδοχέα σύμφωνα με την πολιτική που ακολουθούμε, τις αντιστοιχεί και στέλνει στους υποδοχείς τα κατάλληλα μηνύματα

4.3. Πειραματική Αξιολόγηση

Πειραματική Διάταξη. Οι δοκιμές έγιναν σε τρεις εικονικές μηχανές που φιλοξενήθηκαν στο okeanos του grnet με 1 cpu core (2Ghz), 4 GB RAM και η ταχύτητα του δικτύου που τις ενώνει κυμαίνεται από 10 μεχρι 15mb/sec.

Για να αξιολογήσουμε την απόδοση του δυναμικού συστήματος έπρεπε να δημιουργήσουμε ερώτημα που να έχει απρόβλεπτη συμπεριφορά.

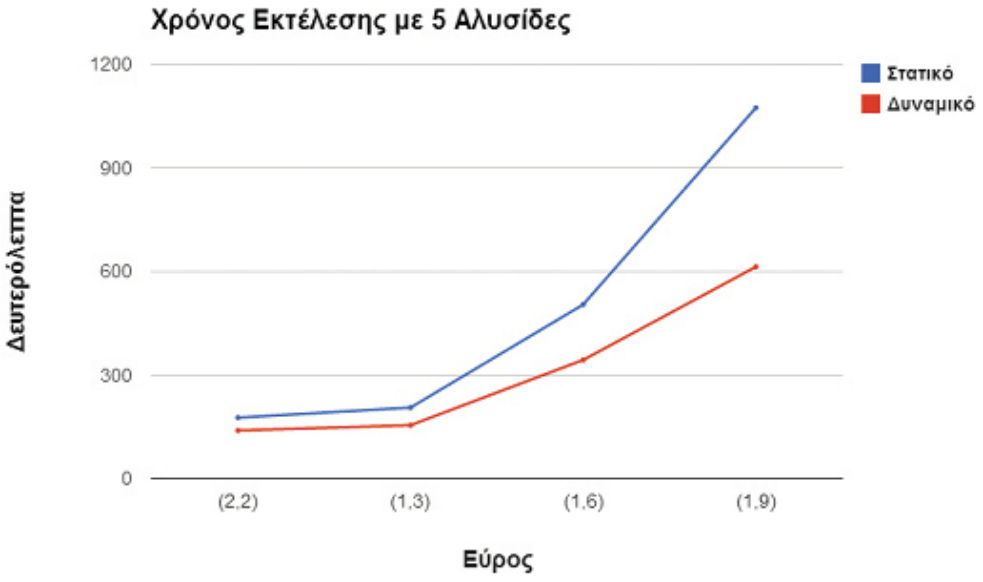
Πρώτα δημιουργήσαμε την randcompute (min,max) μια row συνάρτηση που κάθε φορά που καλείται υπολογίζει τυχαίο πλήθος ψηφίων του π μεταξύ min και max. Το ερώτημα χωρίζεται σε λογικά στάδια, σε κάθε στάδιο έχει πολλούς ανεξάρτητους τελεστές. Κάναμε αρκετές δοκιμές αλλάζοντας το εύρος της τυχαιότητας.

Μετά δημιουργήσαμε ερωτήματα που οι τελεστές του επενεργούν σε μεγάλο όγκο δεδομένων και έχουν δενδρική μορφή. Το ερώτημα χωρίζει τον αρχικό πίνακα σε partition και στη συνέχεια τον κάνει merge δενδρικά.

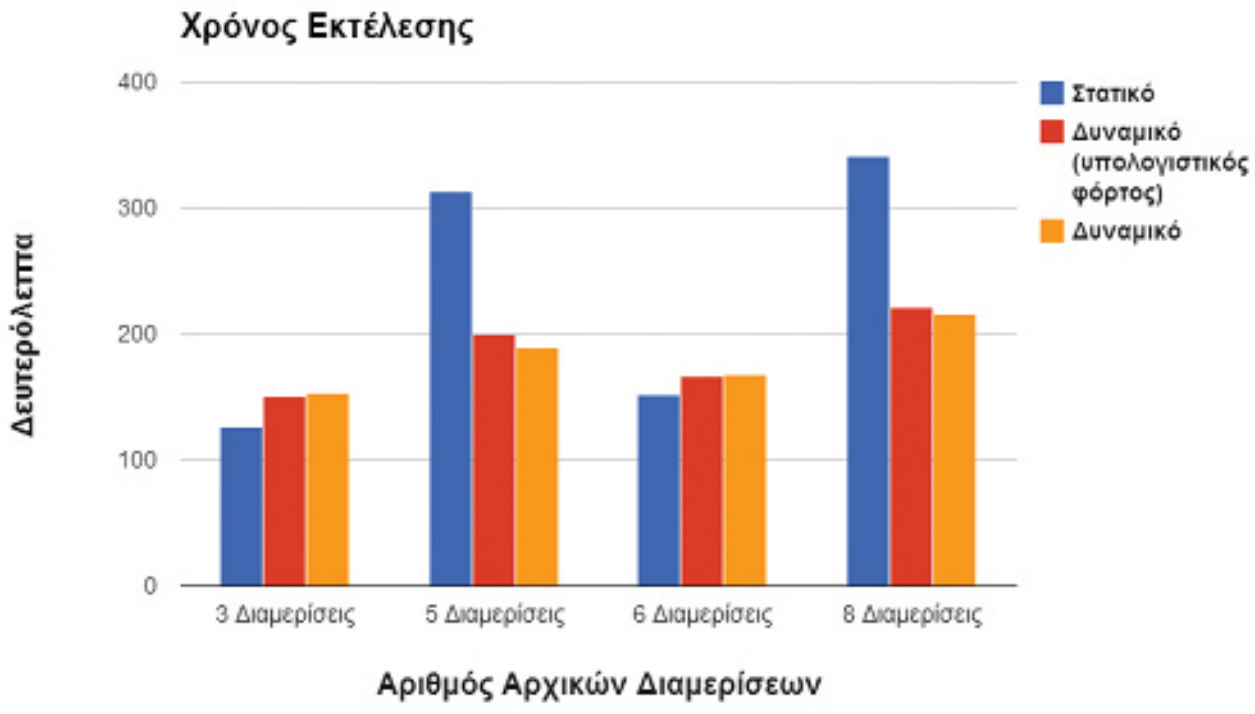
Τέλος δημιουργήσαμε ερωτήματα με μεγάλες αποκλίσεις στον προβλεπόμενο και τον πραγματικό χρόνο εκτέλεσης των τελεστών.

Μετρικές. Χρόνος Εκτέλεσης : Ο χρόνος από τη στιγμή που θα γίνει υποβολή του πλάνου μέχρι και το τερματισμού του τελευταίου υποδοχέα.

Αποτελέσματα.



Σχήμα 4: Χρόνος εκτέλεσης με 5 αλυσίδες σε σχέση με το εύρος τυχαιότητας των υπολογισμών.



Σχήμα 5: Σύγκριση χρόνου εκτέλεσης σε σχέση με το αρχικό πλήθος των διαμερίσεων

Ερώτημα με randcompute. (Σχήμα 4) Ο οριζόντιος άξονας αναπαριστά το εύρος της τυχαιότητας των υπολογισμών που κάνει κάθε τελεστής, στην περίπτωση αυτή εκατοντάδες ψηφία του π που υπολογίζονται. Κάθε αλυσίδα είναι μια ομάδα ερωτημάτων που έχουν μόνο εσωτερικές εξαρτήσεις και εκτελούνται σειριακά. Οι αποφάσεις για την αντιστοίχιση ομάδας-υποδοχέα βασίζονται μόνο στο φόρτο εργασίας κάθε υποδοχέα.

Οι αρχικοί πίνακες του δεντρικού ερωτήματος είναι χωρισμένοι με βάση το κλειδί. (Σχήμα 5). Τρέξαμε το ερώτημα που χωρίζει σε 3, 5, 6, 8 διαμερίσεις τον αρχικό πίνακα με βάση το ίδιο κλειδί και μετά κάνει δεντρικά ένωση ανά 2. Παρατηρούμε μεγάλη βελτίωση στις περιπτώσεις που οι αρχικές διαμερίσεις του πίνακα δεν είναι πολλαπλάσια των μηχανημάτων.

4.4. Συμπέρασμα

Στις δοκιμές που χρησιμοποιούσαμε στατιστικά μόνο για τον φόρτο κάθε υπο-

δοχέα και τα ερωτήματα αποτελούνταν από στάδια με τελεστές χωρίς εξαρτήσεις και κάθε τελεστής εκτελούσε τυχαίους σε πλήθος υπολογισμούς, τα αποτελέσματα που λάβαμε ήταν θετικά. Η βελτίωση στο χρόνο εκτέλεσης σε σχέση με το στατικό χρονοπρογραμματισμό αυξάνεται όσο αυξάνει το δυνατό πλήθος των τυχαίων υπολογισμών.

Στόχος του δυναμικού συστήματος είναι να αντιμετωπίσει τις περιπτώσεις που μη προβλέψιμα γεγονότα επηρεάζουν την απόδοση του συστήματος. Αυτά τα γεγονότα μπορεί να οφείλονται είτε σε λάθος εκτιμήσεις του συστήματος είτε σε εξωτερικούς παράγοντες, σε αυτές τις περιπτώσεις ο δυναμικός χρονοπρογραμματισμός έχει θετικά αποτελέσματα και προσφέρει βελτίωση.

Σε δοκιμές χωρίς τέτοια γεγονότα παρατηρήσαμε ότι το δυναμικό σύστημα πετυχαίνει παραπλήσιους χρόνους με τα στατικό σύστημα, το πρόβλημα παρουσιάζεται όταν τα δεδομένα που απαιτεί το ερώτημα ξεπεράσουν κάποιο όριο. Πιστεύουμε ότι με πιο αποδοτική διαχείριση των μεταφορών των δεδομένων το δυναμικό σύστημα μπορεί να γίνει πιο γρήγορο ή να πετυχαίνει τους ίδιους χρόνους σε όλες τις περιπτώσεις.

ΑΝΑΦΟΡΕΣ

[8] Manolis M. Tsangaris, George Kakaletris, Herald Kllapi, Giorgos Papanikos, Fragkiskos Pentaris, Paul Polydoras, Eva Sitaridi, Vassilis Stoumpos, Yannis E. Ioannidis: Dataflow Processing and Optimization on Grid and Cloud Infrastructures. IEEE Data Eng. Bull. 32(1): 67-74 (2009)

[9] http://doc.madis.googlecode.com/hg/index.html

[10] K. Ousterhout, P. Wendell, M. Zaharia, I. Stoica. “Sparrow: distributed, low latency scheduling”, in Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles, SOSP’13, pp. 69-84.

[11] M. Mitzenmacher. “The Power of Two Choices in Randomized Load Balancing”. IEEE Transactions on Parallel and Distributed Computing, 12(10):1094–1104, 2001.

Επεξεργασία Ρευμάτων Δεδομένων σε Υπολογιστικό Νέφος

Χριστόφορος Σβίγγος (c.sviggos@di.uoa.gr)

Περίληψη

Η ανάγκη επίλυσης προβλημάτων όπως αυτά προέκυψαν από την ραγδαία εξάπλωση των κοινωνικών δικτύων και των συναλλαγών υψηλής συχνότητας, έφερε την ανάγκη ανάπτυξης κατανεμημένων συστημάτων επεξεργασίας ροών δεδομένων σε πραγματικό χρόνο. Σε αυτήν την εργασία παρουσιάζεται η ανάπτυξη ενός συστήματος επεξεργασίας ροών δεδομένων σε υπολογιστικό νέφος. Για την ανάπτυξη του συστήματος χρησιμοποιήθηκε το προϋπάρχον ADP (Athena Distributed Processing) σύστημα, που δημιουργήθηκε στο Τμήμα Πληροφορικής και Τηλεπικοινωνιών, και προσφέρει ελαστική επεξεργασία δεδομένων σε συστάδες υπολογιστών. Το σύστημα επεξεργασίας ροών που παρουσιάζεται, προσφέρει μία εύρωστη γλώσσα ανάπτυξης ερωτημάτων για την συνεχή επεξεργασία ροών σε κατανεμημένο περιβάλλον. Επίσης, προσφέρει ελαστικότητα στην δέσμευση μηχανημάτων, προσαρμόζοντας τους διαθέσιμους πόρους που έχει στην διάθεσή του συνεχώς, ανάλογα με τη χρονική μεταβολή στα χαρακτηριστικά της ροής που επεξεργάζεται. Για την αξιολόγηση του συστήματος έγιναν πειράματα τα οποία δείχνουν τόσο την ελαστικότητα όσο και την κλιμάκωση του συστήματος, το οποίο είναι σε θέση να επεξεργάζεται δεδομένα σε υψηλούς ρυθμούς.

Λέξεις κλειδιά: Σύστημα Επεξεργασίας Ρευμάτων Δεδομένων, Ρεύμα Δεδομένων, Υπολογιστικό Νέφος, SQL

Επιβλέποντες

Ιωάννης Ιωαννίδης, Καθηγητής, Χεράλντ Κλάπι, Υποψήφιος Διδάκτωρ

1. Εισαγωγή

Τα τελευταία χρόνια υπήρξε μεγάλο ενδιαφέρον όσο αφορά την έρευνα και την ανάπτυξη συστημάτων επεξεργασίας ρευμάτων δεδομένων σε πραγματικό χρόνο [4], [5], [6], [7], [8]. Η ερευνητική αυτή δραστηριότητα προήλθε από την ανάγκη επίλυσης προβλημάτων όπως παρακολούθηση της δραστηριότητας δικτύου (network monitoring), ανάλυση οικονομικών δεδομένων καθώς και δίκτυα αισθητήρων τα οποία απαιτούν την συνεχή εκτέλεση συγκεκριμένων ερωτημάτων πάνω σε συνεχή απεριόριστα δεδομένα. Η ανάγκη επίλυσης νέων προβλημάτων, όπως αυτά προέκυψαν από την ραγδαία εξάπλωση των κοινωνικών δικτύων και των συναλλαγών υψηλής συχνότητας (high-frequency trading), καθώς και η ραγδαία εξέλιξη των κατανεμημένων συστημάτων μεγαλώνει την ανάγκη συνέχισης της έρευνας συστημάτων επεξεργασίας ρευμάτων δεδομένων.

Ένα μεγάλο ζήτημα όσο αφορά την επεξεργασία ρευμάτων είναι τα ίδια τα δεδομένα τα οποία είναι συνεχή απεριόριστα και χρονικά μεταβαλλόμενα. Τα συστήματα επεξεργασίας ρευμάτων διαχειρίζονται δεδομένα, πάνω στα οποία δεν έχουν κανένα έλεγχο όσο αφορά τον ρυθμό μετάδοσης τους. Η εκτέλεση συνεχών ερωτημάτων πάνω σε τέτοια δεδομένα, έχει σαν αποτέλεσμα να προσδίδουν μεγάλο φόρτο εργασίας τόσο σε επεξεργαστική ισχύει και μνήμη όσο και σε εύρος ζώνης δικτύου. Δεδομένου ότι ο φόρτος εργασίας μπορεί να ποικίλει με απρόβλεπτους τρόπους, τα συστήματα αυτά πρέπει να είναι ικανά να προσαρμόζονται τόσο στα δεδομένα όσο και στον ρυθμό εισροής τους, αλλά και να εκμεταλλεύονται όσο το δυνατό καλύτερα τους περιορισμένους πόρους που διαθέτουν.

Σκοπός της δικιά μας προσέγγισης είναι η ανάπτυξη ενός κατανεμημένου συστήματος επεξεργασίας ρευμάτων δεδομένων, με στόχο το υπολογιστικό νέφος (cloud computing), το οποίο από την μία θα παρέχει μια εύρωστη γλώσσα ανάπτυξης ερωτημάτων βασισμένη στα πρότυπα της SQL, και από την άλλη θα έχει την ικανότητα να προσαρμόζεται κατάλληλα με το ρεύμα δεδομένων ώστε να παρέχει όσο το δυνατό ελάχιστο “κόστος”. Με τον όρο κόστος αναφερόμαστε στους πόρους που χρειάζεται να δεσμευτούν για την επεξεργασία των δεδομένων. Για την ανάπτυξη του συστήματος μας βασίστηκε πάνω στο προϋπάρχον σύστημα ADP [2] το οποίο παρέχει κατανεμημένοι επεξεργασία πάνω σε στατικά δεδομένα. Βασική ιδέα του συστήματος μας είναι η τμημα-

τοποίηση σειριακά του ρεύματος δεδομένων βάσει μιας χρονικής οντότητας που ονομάζεται κβάντο. Τα συνεχή ερωτήματα πραγματοποιούνται πάνω σε δέσμες δεδομένων τα οποία έχουν τμηματοποιηθεί βάσει του κβάντου.

Σύμφωνα με αυτά που γνωρίζουμε, δεν υπάρχει άλλο σύστημα επεξεργασίας ρευμάτων δεδομένων το οποίο να προσφέρει τη δυνατότητα ελαστικής δέσμευσης μηχανημάτων στο υπολογιστικό νέφος καθώς και τη δυνατότητα δηλώσης συνεχών ερωτημάτων στο πρότυπο της SQL. Από την μεριά μας προτείνουμε μια πολύ καθαρή και εύρωστη γλώσσα ανάπτυξης συνεχών ερωτημάτων κάτι το οποίο φαίνεται να εκλείπει από πολλά σύγχρονα συστήματα. Αρκετά συστήματα επεξεργασίας ρευμάτων, κατευθύνονται προς ένα προγραμματιστικό μοντέλο ανάπτυξης ερωτημάτων, το οποίο ξεφεύγει από τα πρότυπα της SQL. Επιπρόσθετα, το σύστημα μας προσαρμόζεται στο ρεύμα δεδομένων προσπαθώντας και πετυχαίνοντας με αρκετά ενθαρρυντικά αποτελέσματα να χρησιμοποιεί όσα μηχανήματα χρειάζεται μια εκάστοτε χρονική στιγμή για μια συγκεκριμένη ταχύτητα μετάδοσης δεδομένων. Όσον αφορά την απόδοση του συστήματός μας, είναι αρκετά ικανοποιητική όπως φαίνεται από τα πειράματα.

2. Ανασκόπηση Συστήματος

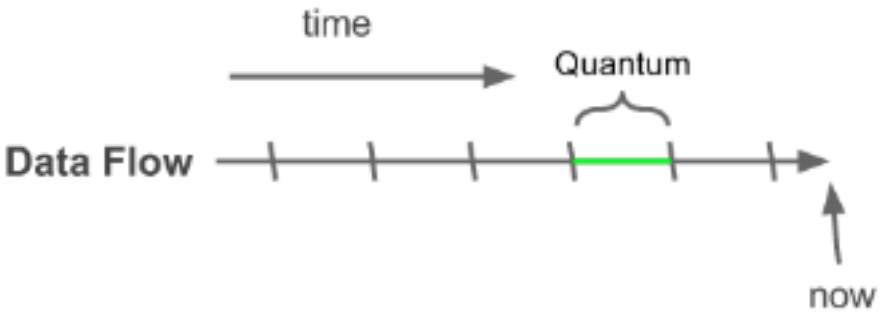
Το σύστημά μας, χτίστηκε πάνω από ένα στατικό σύστημα βάσεων δεδομένων, εκείνο του ADP. Το ADP αποτελεί ένα σύστημα αποθήκευσης και επεξεργασίας μεγάλου όγκου δεδομένων χρησιμοποιώντας συστάδες υπολογιστών (clusters).

Πρόκληση αποτέλεσε η μετατροπή ενός τέτοιου συστήματος, το οποίο είναι προορισμένο να επεξεργάζεται στατικά δεδομένα, σε ένα σύστημα επεξεργασίας ρευμάτων δεδομένων.

2.1. Κβάντα και Ροές

Για τη μετατροπή ενός στατικού συστήματος βάσεων δεδομένων, σε ένα σύστημα επεξεργασίας ροών, είναι απαραίτητη η τμηματοποίηση των ροών. Με την τμηματοποίηση των ροών, μετατρέπουμε τα απεριόριστα δεδομένα μιας συνεχούς ροής σε απεριόριστα τμήματα, τα οποία περιέχουν πεπερασμένα

δεδομένα. Αυτό μας δίνει τη δυνατότητα επεξεργασίας του κάθε τμήματος ξεχωριστά από στατικά συστήματα βάσεων δεδομένων, καθώς και τη δυνατότητα ανάπτυξης συνεχών ερωτημάτων τα οποία εκτελούνται περιοδικά σε κάθε νέο τμήμα.



Σχήμα 1: Τμηματοποίηση ρεύματος δεδομένων σε κβάντα

Στο σύστημά μας, ως μονάδα κατάτμησης της ροής δεδομένων ορίσαμε το κβάντο. Το κβάντο είναι μια προκαθορισμένη χρονική μονάδα, βάσει της οποίας η ροή δεδομένων "τμηματοποιείται" σε μη επικαλυπτόμενα χρονικά παράθυρα ίσου μεγέθους. Το κβάντο ορίζει ένα στατικό παράθυρο. Κάθε νέο κβάντο δημιουργείται εφόσον περάσει χρόνος ίσος με μία Κβαντική μονάδα από τη δημιουργία του τελευταίου κβάντου όπως φαίνεται στο Σχήμα 1.

2.2. Γλώσσα

Θεμελιώδης στοιχείο της γλώσσας μας είναι το κβάντο. Το κβάντο είναι καθολικό για όλο το σύστημα. Όλα τα ρεύματα τμηματοποιούνται βάσει του ίδιου κβάντου, και είναι συγχρονισμένα με αυτό. Με τον όρο "συγχρονισμένα", εννοούμε ότι τα κβάντα κάθε ρεύματος είναι χρονικά συγχρονισμένα με τα κβάντα όλων των άλλων ρευμάτων. Έτσι όταν αναφερόμαστε στο N κβάντο ξέρουμε πως αναφερόμαστε σε όλες τις Πλειάδες από το χρόνο $(i, j]$ όπου $i<j$ και i, j σταθερές.

Η γλώσσα μας επέκτεινε καταλλήλως το μοντέλο του ADP ώστε να υποστηρίζει την επεξεργασία ρευμάτων δεδομένων. Για το λόγο αυτό εισήχθηκε ένας νέος τύπος, εκείνος του stream. Πιο συγκεκριμένα η γλώσσα μας υποστηρίζει

δηλώσεις τις μορφής:

```
Distributed Create Stream <streamname> As
ExtendedADP-QLStatement ...
```

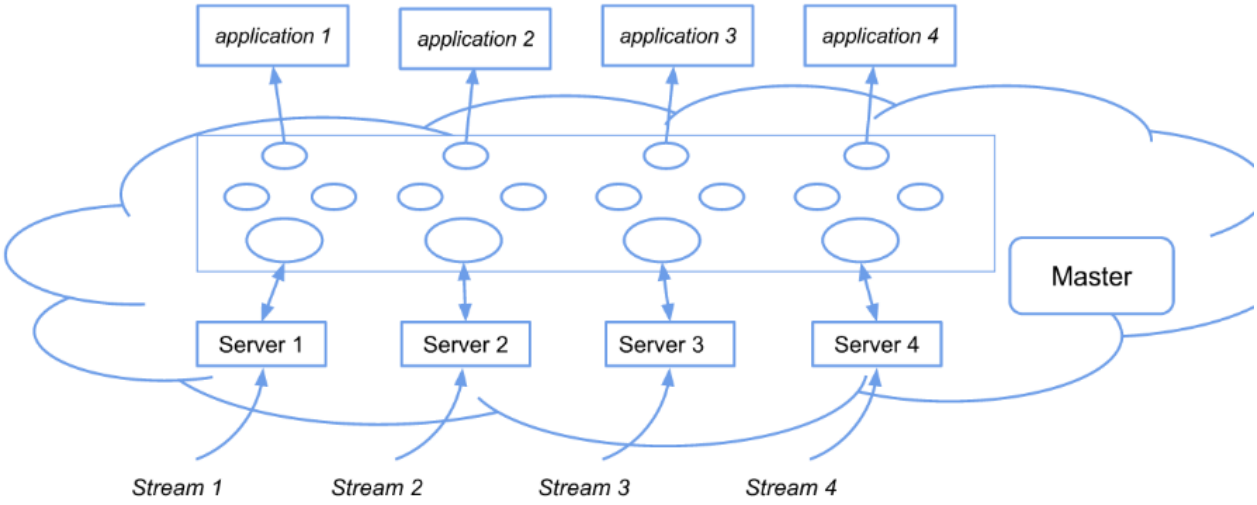
η οποία διαισθητικά, εκτελεί ένα συνεχές ερώτημα πάνω στις ροές δεδομένων, σε ένα κατανεμημένο περιβάλλον. Το άλλο βασικό στοιχείο της γλώσσας μας έχει να κάνει με τα κβάντα, και πώς μπορούμε να αναφερθούμε σε αυτά. Όπως είπαμε, κάθε ροή είναι τμηματοποιημένη σε μη επικαλυπτόμενα κβάντα. Στην γλώσσα μας η αναφορά σε κάποιο κβάντο γίνεται με την προσθήκη της συμβολοσειράς [<Quantum Number>] δίπλα από την δήλωση κάθε ονόματος ροής στην δήλωση From, όπως φαίνεται παρακάτω.

```
Distributed Create Stream <Stream Name> As
Select *
From <Another Stream Name>[<Quantum Number>];
```

Η συμβολοσειρά [<Quantum Number>], δηλώνει το πλήθος των κβάντων πριν το τελευταίο κβάντο της ροής. Η αρίθμηση των κβάντων ξεκινάει από το μηδέν. Έτσι με τη δήλωση stream_1[0], παίρνουμε το τελευταίο κβάντο της ροής με όνομα stream_1. Για να πάρουμε το κβάντο που αναφέρεται σε δεδομένα τα οποία ήρθαν 10 κβάντα πριν από το τελευταίο κβάντο στην ροή με όνομα stream_1, αναφερόμαστε με τη δήλωση stream_1[10].

2.3. Αρχιτεκτονική

Στην προηγούμενη ενότητα ορίσαμε το τι ακριβώς είναι το κβάντο, και πώς αυτό τμηματοποιεί τα ρεύματα δεδομένων. Τώρα είμαστε έτοιμοι να δούμε τη συνολική αρχιτεκτονική του συστήματος, όπως αυτή απεικονίζεται στο σχήμα 2. Το σύστημά μας, το οποίο είναι ένα σύστημα κατανεμημένης επεξεργασίας δεδομένων, αποτελείται από μια συστάδα υπολογιστών. Μέσα στη συστάδα υπολογιστών, μπορούμε να ξεχωρίσουμε τριών ειδών κόμβους. Κάθε κόμβος, διαχωρίζεται από τους υπόλοιπους με βάση τις διαφορετικές εργασίες που εκτελεί. Έτσι έχουμε τους εξυπηρετητές (servers), οι οποίοι επικοινωνούν με τις εξωτερικές ροές δεδομένων. Και τέλος τον κόμβο αφέντη (master), ο οποίος είναι υπεύθυνος, για την κατανομή εργασιών στους εργάτες.



Σχήμα 2: Αρχιτεκτονική του συστήματος

2.4. Προσαρμοστικός Αλγόριθμος Δέσμευσης Μηχανημάτων

Τα ρεύματα δεδομένων έχουν το χαρακτηριστικό ότι είναι συνεχής και χρονικά μεταβαλλόμενες. Εφόσον έχουμε να κάνουμε με δεδομένα στα οποία ο ρυθμός άφιξης στο σύστημα είναι απρόβλεπτος και συνεχώς μεταβαλλόμενος, θα ήταν ιδανικό να χρησιμοποιούνται, σε κάθε χρονική στιγμή τόσα μηχανήματα όσα ακριβώς χρειάζονται για την επεξεργασία των δεδομένων. Έτσι, σε χρονικές περιόδους κατά τις οποίες ο ρυθμός άφιξης δεδομένων θα είναι πολύ μικρός, θα χρησιμοποιείται ένα μικρό υποσύνολο των συνολικών μηχανημάτων της συστάδας υπολογιστών. Αυτό θα είχε ως αποτέλεσμα τη δραστική μείωση του κόστους επεξεργασίας των δεδομένων.

Το παραπάνω πρόβλημα, παρά της σημασίας του και της δυσκολίας επίλυσης του, δεν έχει απασχολήσει ακόμα τα άλλα συστήματα κατανεμημένης επεξεργασίας ρευμάτων. Στο σύστημά μας τα δεδομένα πρώτα τμηματοποιούνται σε κβάντα και ύστερα γίνεται η επεξεργασία τους. Αυτό μας δίνει την ευελιξία να εφαρμόσουμε αλγορίθμους η οποίοι μπορούν να προβλέψουν και να προσαρμόσουν την δέσμευση μηχανημάτων για την επεξεργασία του τρέχοντος κβάντου, παίρνοντας στατιστικά από την επεξεργασία του προηγούμενου. Παρόμοια προβλήματα, όπως η προσαρμογή της αποστολής TCP/IP δεδομενο-

γραμμάτων, ώστε να αποφεύγεται η συμφόρηση στο δίκτυο, έχουν απασχολήσει τους ερευνητές δικτύων.

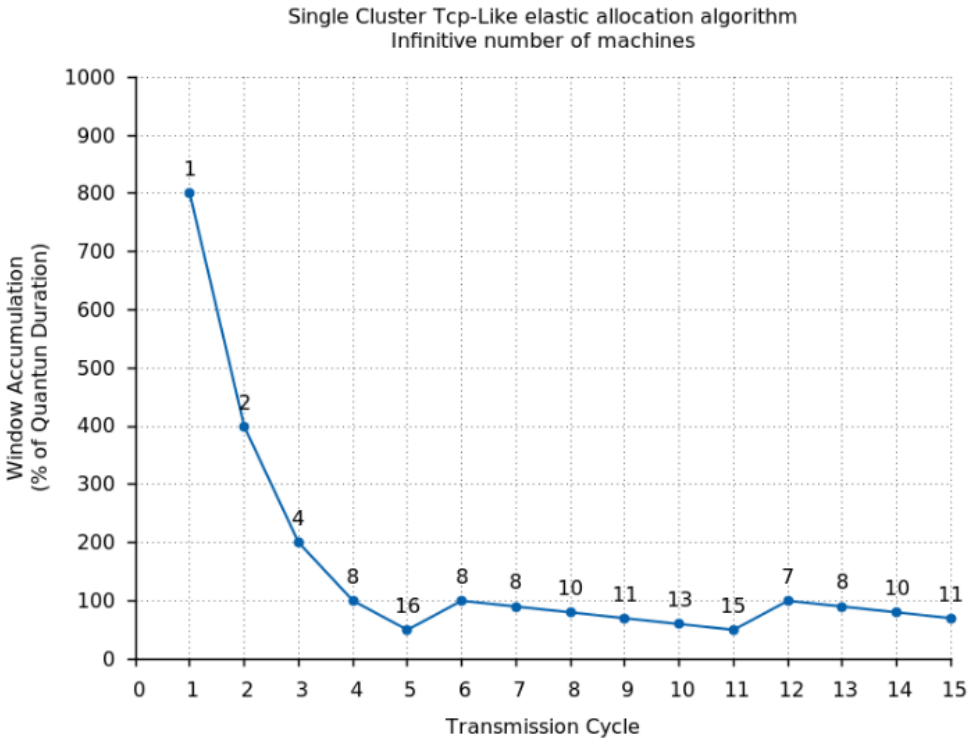
Ένας τέτοιος αλγόριθμος είναι ο TCP-Reno [9], τον οποίο προσαρμόσαμε και εμείς για τις ανάγκες του προβλήματος στο σύστημά μας, ώστε να δεσμεύουμε το ιδανικό πλήθος μηχανημάτων. Ως ιδανικό πλήθος ορίζεται η ελάχιστη ποσότητα μηχανημάτων που πρέπει να δεσμευτεί, ώστε ο χρόνος που απαιτείται για την επεξεργασία του κβάντου από το σύνολο των συνεχών ερωτημάτων, να είναι μικρότερος από τη χρονική ποσότητα που ορίζει το κβάντο. Για να πετύχουμε το συγκεκριμένο στόχο, υλοποιήσαμε τον TCP-Reno αλγόριθμο, προσαρμοσμένο στις απαιτήσεις του προβλήματός μας. Ποιο συγκεκριμένα, στην αρχή δεσμεύεται ένα μηχάνημα και υπολογίζεται ο λόγος του συνολικού χρόνου επεξεργασίας του κβάντου προς τον χρόνο που ορίζει το κβάντο. Ο λόγος αυτός αποτελεί το παράθυρο, ενώ ως κύκλος μετάδοσης ορίζεται ο αριθμός του κβάντου που επεξεργάζεται από το σύστημα. Έτσι έχουμε ότι:

παράθυρο = Συνολικός Χρόνος Επεξεργασίας Κβάντου / Χρόνος Κβάντου

Στη συνέχεια μειώνουμε το παράθυρο στο μισό και υπολογίζουμε τον αριθμό μηχανημάτων που πρέπει να δεσμευτούν, ώστε να πετύχουμε το στόχο που ορίζεται από το νέο παράθυρο. Ο υπολογισμός των μηχανημάτων προκύπτει από την εξίσωση:

μηχανήματα(ν) = (παράθυρο(ν) / παράθυρο(ν-1)) * μηχανήματα(ν-1),

όπου ν είναι ο κύκλος μετάδοσης. Η διαδικασία αυτή είναι παρόμοια με τη διαδικασία που ακολουθεί ο TCP-Reno αλγόριθμος κατά τη διάρκεια της αργής εκκίνησης. Η περίοδος αυτή, η οποία απεικονίζεται στην εικόνα 11 κατά τους κύκλους μετάδοσης 1 έως 5, συνεχίζεται έως ότου έχουμε μια κατάσταση σύγκρουσης.



Σχήμα 3: Αναπαράσταση λειτουργίας του TCP αλγορίθμου για Cluster

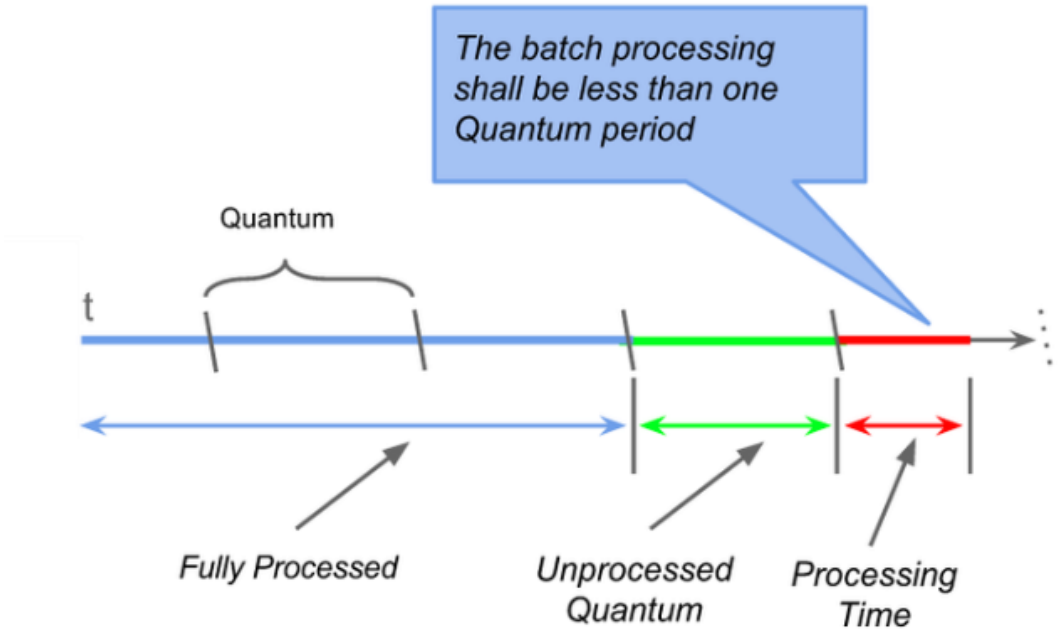
Κατάσταση σύγκρουσης έχουμε όταν το μέγεθος του παράθυρου γίνει μικρότερο από το μισό του χρόνο που ορίζει το κβάντο ή όταν ο αριθμός των μηχανημάτων που πρέπει να δεσμεύσουμε είναι μεγαλύτερος από το συνολικό πλήθος μηχανημάτων στη συστάδα υπολογιστών. Σε τέτοιες καταστάσεις, το παράθυρο διπλασιάζεται, και υπολογίζεται εκ νέου ο αριθμός μηχανημάτων, όπως φαίνεται στους κύκλους μετάδοσης 6, 12. Μετά από μία κατάσταση σύγκρουσης περνάμε σε μια κατάσταση προσθετικής μείωσης κατά τη διάρκεια της οποίας το παράθυρο μειώνεται κατά μια μονάδα της τάξης του 0.1.

3. Πειραματική Αξιολόγηση

3.1. Ρυθμίσεις Συστήματος

Για την αξιολόγηση του συστήματος μας, είχαμε στη διάθεση μας τέσσερις συνολικά υπολογιστές. Κάθε υπολογιστής ήταν εφοδιασμένος με επεξεργαστή Intel Xeon E5620 ο οποίος έχει δύο συνολικά πυρήνες χρονισμένους στα 2.4 GHz, καθώς και 4 GB μνήμη RAM. Για τα πειράματα μας χρησιμοποιήσαμε μια ροή

δεδομένων η οποία αποτελούνταν από πέντε πεδία τα οποία αποθηκεύουν πέντε ακεραίους αντίστοιχα. Για τους πειραματικούς μας σκοπούς, η ροή παραγόταν τυχαία χωρίς να αναπαριστά κάποιο ρεαλιστικό παράδειγμα.



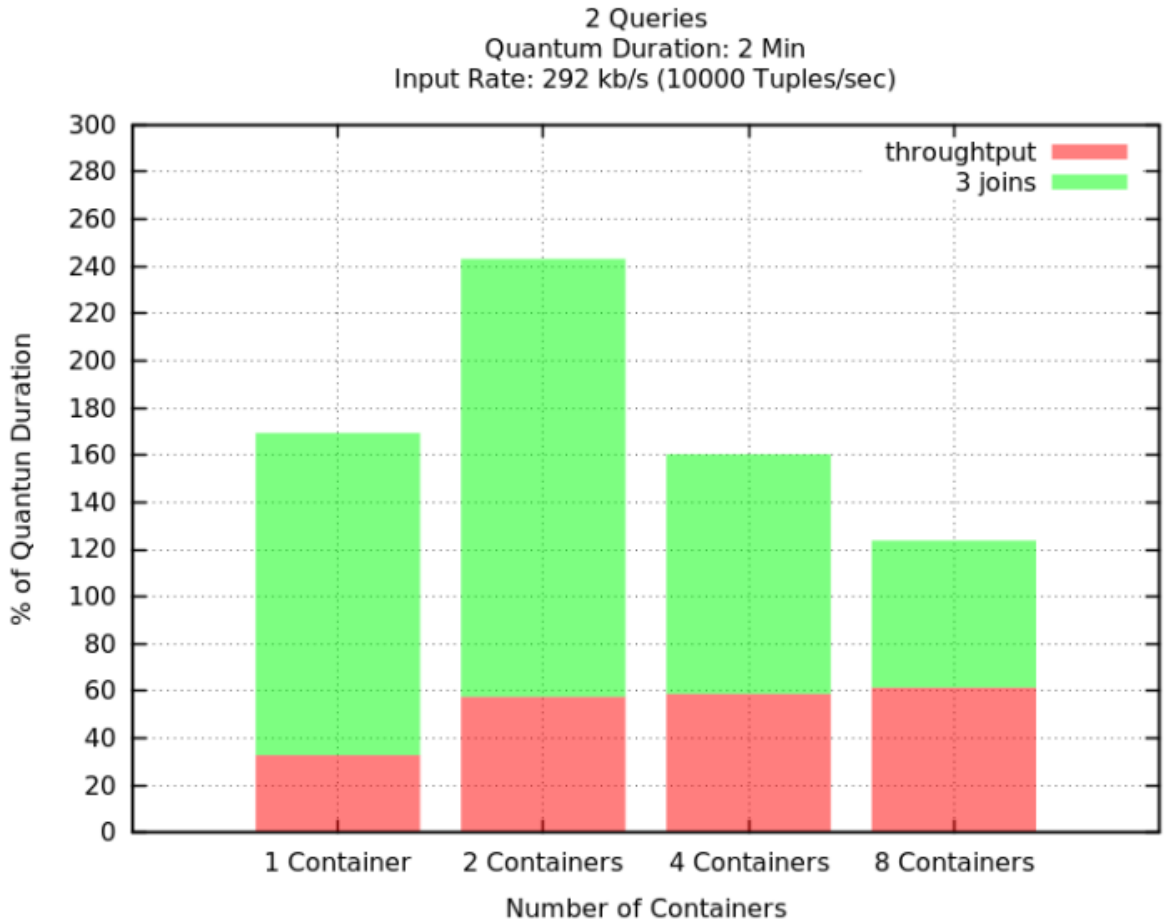
Σχήμα 4: Στόχος για την ομαλή επεξεργασία των δεδομένων

Στα πειράματα μας μελετάμε το σύστημά μας τόσο ως προς το ρυθμό επεξεργασίας δεδομένων όσο και ως προς τη συμπεριφορά του σε διαφορετικές συνθήκες λειτουργίας, όπως διαφορετικοί τύποι ερωτημάτων, διαφορετικοί χρονικοί περίοδοι για το κβάντο, και διαφορετικό πλήθος μηχανημάτων. Επίσης αξιολογήσαμε και την ελαστικότητα στη δέσμευση μηχανημάτων του συστήματός μας, σε συνθήκες μεταβολής του ρυθμού εισροής δεδομένων.

Στόχος για την ομαλή επεξεργασία των δεδομένων μας είναι η επεξεργασία του κάθε κβάντου να διαρκεί χρόνο ίσο ή μικρότερο από τη χρονική περίοδο που ορίζεται από ένα κβάντο, όπως φαίνεται στο σχήμα 4. Για το λόγο αυτό στα διαγράμματα που ακολουθούν, ο κάθετος άξονας αναπαριστά το λόγο του συνολικού χρόνου επεξεργασίας του κβάντου ως προς τη χρονική περίοδο που ορίζει ένα κβάντο.

3.2. Κλιμάκωση

Στο διάγραμμα 6,που ακολουθεί, απεικονίζεται οι συμπεριφορά του συστήματος, όταν σαν είσοδο πάρει δύο συνεχή ερωτήματα όπου η έξοδος του ενός είναι η είσοδος του άλλου. Ποιο αναλυτικά το ερώτημα με όνομα “throughput”, είναι ένα ερώτημα επικοινωνίας με τον εξυπηρετητή, για την εισροή εξωτερικών δεδομένων, ενώ το ερώτημα με όνομα “3 joins” είναι ένα ερώτημα στο οποίο πραγματοποιούνται τρεις ζεύξεις ισότητας. Στον οριζόντιο άξονα απεικονίζεται ο αριθμός των containers που χρησιμοποιήθηκε κάθε φορά για τις μετρήσεις. Να σημειωθεί ότι όλες οι μετρήσεις έγιναν με χρονική περίοδο κβάντου ίση με δύο λεπτά και ρυθμό εισροής δεδομένων ίσο με 292 KB/s.



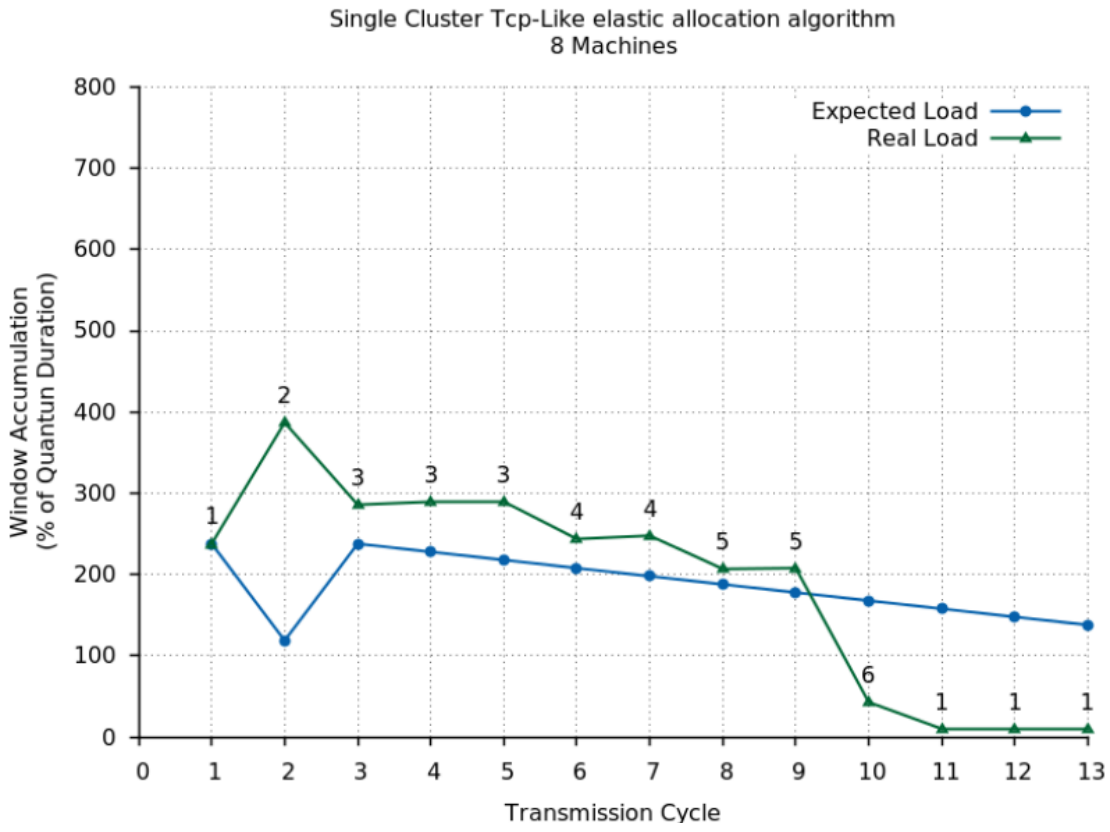
Σχήμα 5: Συμπεριφορά συστήματος για την εκτέλεση ενός γράφου με δύο συνεχή ερωτήματα

Για το ερώτημα “throughput”, ο χρόνος εκτέλεσης αυξάνεται όταν είναι σε λειτουργία δύο ή περισσότεροι container, λόγω του γεγονότος ότι τα δεδο-

μένα εισέρχονται από ένα κόμβο και διασπούνται έπειτα σε περισσότερους. Παρόλα αυτά, όσο αυξάνεται ο αριθμός των εν λειτουργία Container, ο συνολικός χρόνος για την εκτέλεση και των δύο ερωτημάτων μειώνεται. Σε αυτό συνέβαλε η καλή κλιμάκωση που προσφέρεται από το σύστημα, για το ερώτημα "3 Joins".

3.3. Ελαστική Δέσμευση Μηχανημάτων

Για την αξιολόγηση της ελαστικής δέσμευσης των μηχανημάτων, χρησιμοποιήσαμε μια ροή, όπου στους πρώτους εννέα κύκλους μετάδοσης ο ρυθμός εισροής δεδομένων είναι στις δέκα χιλιάδες πλειάδες το δευτερόλεπτο, ενώ στους αμέσως δύο επόμενους κύκλους μετάδοσης ο ρυθμός πέφτει στην μία πλειάδα το δευτερόλεπτο. Αυτό που θα επιθυμούσαμε λοιπόν να δούμε από το σύστημα μας, είναι η απότομη μείωση των μηχανημάτων που δεσμεύονται αμέσως μόλις πέσει ο ρυθμός των δεδομένων στο σύστημα.



Σχήμα 6: Συμπεριφορά συστήματος για την επεξεργασία της ροής που απεικονίζεται στο σχήμα 5

Η συμπεριφορά του συστήματος όταν επεξεργάζεται την ροή που περιγράψαμε φαίνεται στο σχήμα 6. Ποιο αναλυτικά, ο οριζόντιος άξονας αναπαριστά το πλήθος των κύκλων μετάδοσης, ενώ ο κάθετος άξονας αναπαριστά τον γνωστό λόγο του χρόνου της επεξεργασίας ενός κβάντου προς τη χρονική περίοδο του κβάντου. Στο διάγραμμα απεικονίζονται δύο καμπύλες. Η μπλε καμπύλη δείχνει το στόχο που προσπαθεί κάθε φορά να επιτύχει το σύστημα, όπως περιγράφηκε στο αντίστοιχο κεφάλαιο, ενώ η πράσινη καμπύλη δείχνει πώς πραγματικά συμπεριφέρεται το σύστημα.

Η συμπεριφορά λοιπόν, του συστήματός μας είναι αυτή που αναμέναμε. Από τον πρώτο μέχρι τον δέκατο κύκλο, που ο ρυθμός εισροής δεδομένων παραμένει σταθερός στις δέκα χιλιάδες πλειάδες το δευτερόλεπτο, το σύστημά μας προσπαθεί να βρει μια ισορροπία για την επεξεργασία των δεδομένων, αυξάνοντας σιγά σιγά το πλήθος των μηχανημάτων που βρίσκονται σε λειτουργία. Στον δέκατο κύκλο ο ρυθμός εισροής των δεδομένων μειώνεται δραστικά. Τις συνέπειες της μείωσης του ρυθμού εισροής των δεδομένων τον βλέπουμε στον αμέσως επόμενο κύκλο (ενδέκατος κύκλος μετάδοσης), όπου τα μηχανήματα που δεσμεύονται από το σύστημα μας, μειώνονται δραστικά στο ένα.

4. Συμπεράσματα

Η τεχνική του MapReduce [3], άνοιξε την πύλη για τη δημιουργία μιας μεγάλης ποικιλίας συστημάτων κατανεμημένης επεξεργασίας στατικών δεδομένων. Παρόλο της αναγκαιότητας τους, δεν υπάρχει αντίστοιχη δημιουργία συστημάτων επεξεργασίας ρεμάτων δεδομένων στο υπολογιστικό νέφος. Με την εργασία μας δείξαμε ότι κάτι τέτοιο είναι εφικτό χρησιμοποιώντας στατικά συστήματα επεξεργασίας για επεξεργασία ρευμάτων δεδομένων.

Επιπρόσθετα, αναδείξαμε σημαντικά προβλήματα που έχουν να αντιμετωπίσουν τα συστήματα επεξεργασίας ροών, όπως είναι εκείνο της ελαστικής δέσμευσης μηχανημάτων. Όπως έδειξαν τα πειράματα, το σύστημα μας έχει ελαστικότητα για τη δέσμευση μηχανημάτων, καταφέρνοντας να προσαρμόζεται συνεχώς στις μεταβολές του φόρτου επεξεργασίας των ροών δεδομένων.

Η SQL είναι και παραμένει το πρότυπο γλώσσα των συστημάτων βάσεων δεδομένων για πάνω από τρις δεκαετίες, και έχει την ικανότητα να μπορεί να

εκφράζει με απλό τρόπο πολύπλοκους μετασχηματισμούς δεδομένων. Αρκετά σύγχρονα συστήματα [4], [5], [6], έχουν αναπτύξει ένα μοντέλο που απαιτεί προγραμματισμό χαμηλού επιπέδου αποφεύγοντας τη χρήση κάποιας επέκτασης της SQL για την υποστήριξη συνεχών ερωτημάτων. Δυστυχώς ο προγραμματισμός χαμηλού επιπέδου οδηγεί σε μεγάλο χρόνο ανάπτυξης καθώς και σε υψηλό κόστος συντήρησης τέτοιων συστημάτων. Με το σύστημά μας καταφέραμε να ενσωματώσουμε τη δήλωση συνεχών ερωτημάτων σε SQL.

References

[1] Apache. Apache Hadoop, http://hadoop.apache.org

[2] Manolis M. Tsangaris, George Kakaletris, Herald Kllapi, GiorgosPapanikos, Fragkiskos- Pentaris, Paul Polydoras, Eva Sitaridi, VassilisStoumpoos, and Yannis E. Ioannidis, Data f- low Processing and Optimization on Grid and Cloud Infrastructures, IEEE Data Eng, vol. 32, no. 1, Nov. 2009, pp 67-74

[3] James F. Kurose, Keith W. Ross, Computer Networking, 2011, p. 284 J. Dean and S. Ghe- mawat. Mapreduce: simplified data process ing on large clusters. In OSDI'04: Proceedings of the 6th Symposium on Opearting Systems Design & Implementation, pages 10–10, 2004

[4] N. Marz. Twitter storm. https://github.com/nathanmarz/storm/wiki, 2012

[5] Apache S4. http://incubator.apache.org/s4/

[6] T. Akidau, A. Balikov, K. Bekiroglu, S. Chernyak, J. Haberman, R. Lax, S. McVeety, D. Mills, P. Nordstrom, and S. Whittle. MillWheel: Fault-tolerant stream processing at inter- net scale. In VLDB, 2013

[7] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S.Krishnamurthy, S. R. Madden, V. Raman, F. Reiss, and M. A. Shah. TelegraphCQ: Con- tinuous Dataflow Processing for an Uncertain World. In Proc. of the 1st CIDR Conference, Asilomar, CA, 2003.

[8] D. J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a new model and architecture

for data stream manage- ment. The VLDB Journal The International Journal on Very Large Data Bases, 12(2):120–139, 2003.

[9] Kurose & Ross, p. 284

Extracting Interests from Facebook Shares

Stamatis Christoforidis (stachris@di.uoa.gr)

Abstract

Over a billion users worldwide use Facebook to communicate, discover new information, “Like” and “Share” pages. This paper discovers users’ interests from their Shares (focused on Youtube Links), to help them (a) find useful information in the form of Wikipedia articles and (b) expand their profiles with more “Likes”. Our system extracts textual features from each Youtube page and computes relevant topic keywords using LDA. Then, these keywords are matched to the Wikipedia ontology, where further relevant topics are discovered using a novel approach. Finally, we present the results of a user study that measures the accuracy of inferring users interests and the results were promising.

Keywords: Profiling, Recommendations, Facebook, Shares

Advisor

Yannis Ioannidis, Professor, Manos Karvounis, PhD candidate

1. Introduction

Facebook has become the most popular of all social networks (1.11 billion users March 2013)[1]. Facebook users communicate with other users by making friends and doing "Like" to other User's "Shares" (i.e., publish a link from a website they find interesting). In this paper, we analyze users' Youtube Shares (empirically many shares are from Youtube videos). Our purpose is to expand and enrich user's Facebook profiles by suggesting Pages to Like. We also help them discover new information based on their interests by suggesting relevant Wikipedia articles.

The rest of the paper is organized as follows: In Section 2 we present the system's architecture and algorithms, In Section 3 we present the results of a user study. We conclude the paper and discuss future work in Section 4.

2. System Overview

2.1. From Youtube Links to Wikipedia Articles

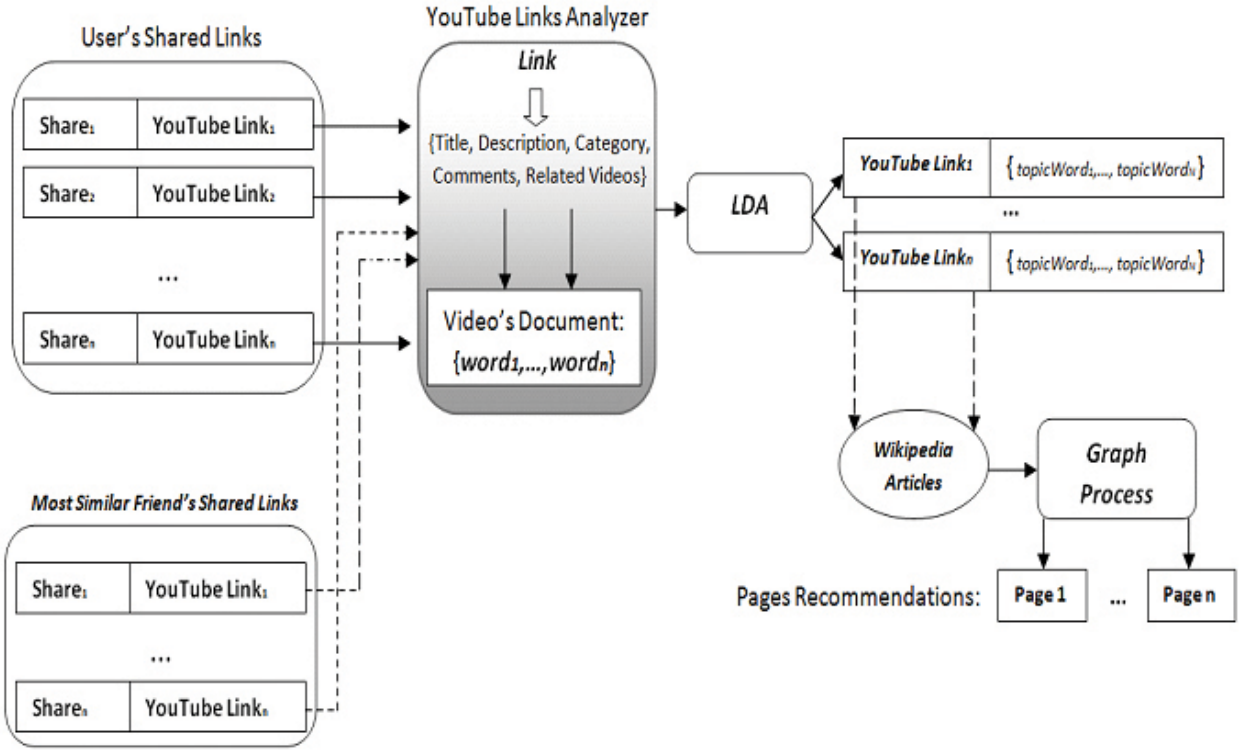
Collecting Shares. For each user, we take the most recent *YT_Count Youtube* Shares and create a set of links that will be processed. If the set is small, we enhance it by adding the Shares of users that are similar to the current user. Similarity is defined as follows: For each friend of the current user we create a Vector with values 0, 1, or -1. Each value is related to a Youtube Share that the current user has interacted with. Value 1 means that the friend of the user has Liked the Youtube Share, -1 that he has published a comment without a Like (a heuristic indication of dislike), and 0 means the friend has not interacted with the Share (neither Liked nor commented). A similar Vector is created for the current user (obviously, it does not have 0 values). We use Cosine Similarity between the Vector of the current user and each of its friends and select the top *K_NumFriends* and add their Youtube Shares to the current user's set of links.

From Youtube to Topics. Given the collection of Youtube Links per user, we extract from each link, (via Youtube V2.0 API) its *Title*, *Category*, *Description*, latest *YT_LinkComments* comments and first *YT_NumRelatedVideos* related Youtube

1 *http://investor.fb.com/releasedetail.cfm?ReleaseID=761090*

videos. Each of these word sequences is split into words to create a Document for each Video. Moreover, the words from Title, Category as from the main Video as from the first *YT_NumRelatedVideos* related Videos, are given an extra weight (by replicating them in the document). We also filter all words with a Stemmer to find those which have the same root and check which words have practically the same meaning. Furthermore, we use the Stanford Log-Linear Part-Of-Speech Tagger[2], to keep only English words from comments that are mostly nouns. Topic words are generally nouns and not verbs or adverbs etc. All these Documents from Youtube links constitute a Corpus. The system uses the Latent Dirichlet Allocation (LDA) [5, 6] with input that Corpus, to generate a set of Words that represent the most likely Topics of the links.

From Topics to Wikipedia. We use Google Custom Search API on every set of topic keywords. We search only for Articles from Wikipedia Pages (source: en.wikipedia.org) and keep the top *Res_GoogleLinks* Wikipedia Links. We now have a set of Wikipedia pages that are relevant to the topics of a user's Shares as produced by LDA.



2 *http://nlp.stanford.edu/downloads/tagger.shtml*

Figure 1: System Overview

2.2. Graph Expansion and Manipulation

Expanding the Graph. For each Google Search response (Wikipedia Article), we collect the first *Wiki_NumLinks* Links from the first Paragraph and then for every one of those, we keep from the first paragraph of the article, only the First Link. We accept, only those links that are (a) Wikipedia articles (b) not inside parenthesis and (c) not special Wikipedia Links such as #Cite, #File etc. We use the resulting Wikipedia articles, to create a directed Graph, where nodes are Wikipedia pages. The edges of the Graph represent the relations between the links in the 1st paragraph of a Wikipedia article. Each child article has links that connect it with parent articles. The parent articles are the links in the first paragraph of a Wikipedia article. Each edge is drawn with direction from the parent article to the child article. For instance, child article "London" (Figure 2) has links that drive to parent articles "England", "United Kingdom", ..., "Medieval". So, there are edges from all the parent articles to child article London. In the same way, article "Country" is connected with the article "England".

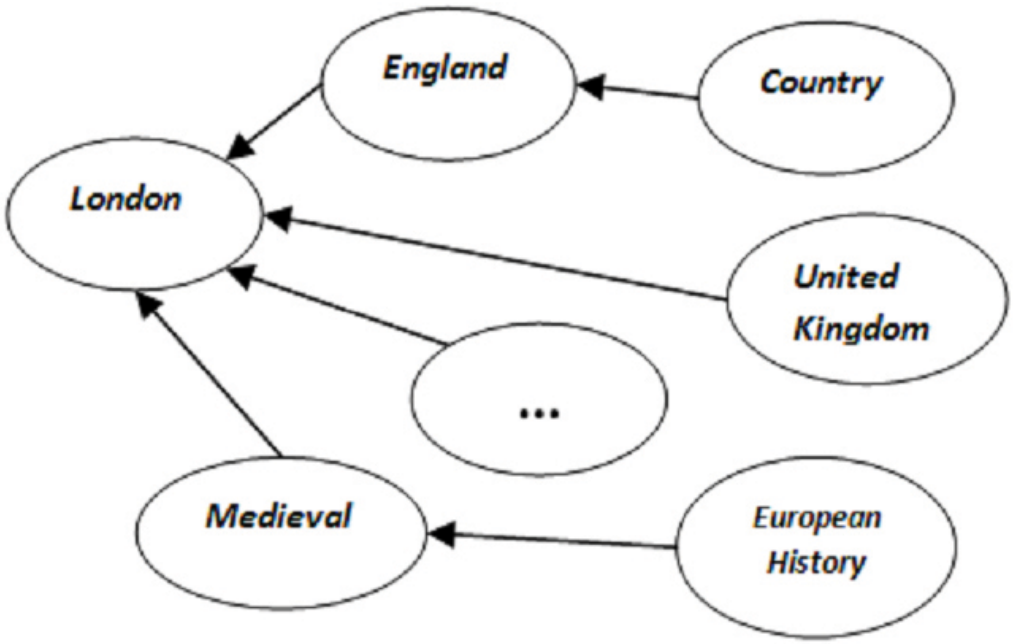


Figure 2: Example of Nodes Relations

Nodes Count. Each node of the Graph has a specific Count:

$$\text{Count} = \text{Immediate\_count} + \text{Derived\_count} \quad (1)$$

Immediate_count: Refers to the occurrences of corresponding Wikipedia article.

Derived_count: It's an extra weight based on the links between a node and its predecessors. This weight is the average count of its parents divided by *w*, plus the average count of its grandparents divided by *2*w*, plus the average count of its great grandparents divided by *3*w*, etc.

At Figure 3 example, suppose there are given (as Google Search results) the articles "Athens" and "Drama, _Greece". Table1 describes the Counts of each node appeared in, with w=6. Notice that node (article) "Greece", appears both in the 1st paragraph of articles "Athens" and "Drama, _Greece", so the occurrences of node "Greece" is 2.

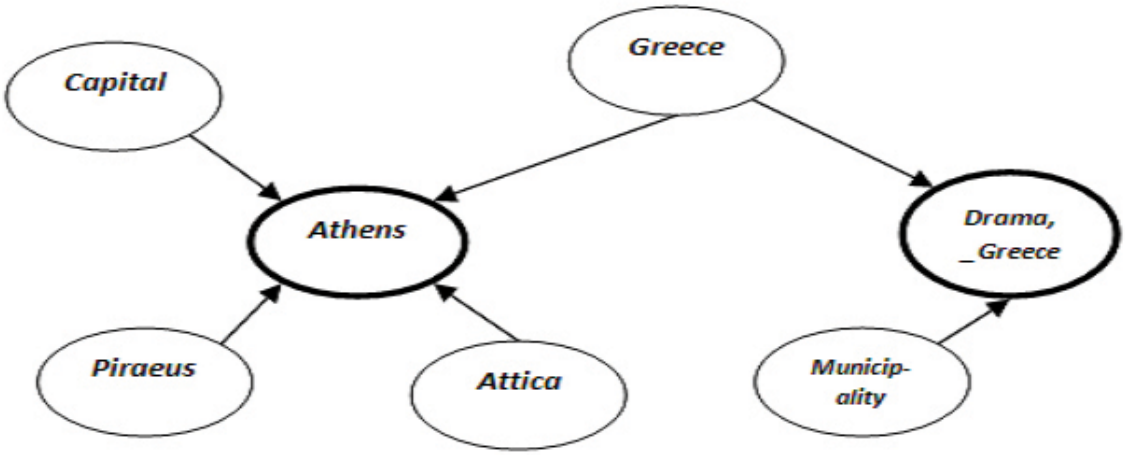


Figure 3: Example of Nodes Count

Table 1: Nodes Count in Figure 3, w=6

Article (Node)	Immediate Count	Derived Count	Total Node Count
Capital	1	-	1
Athens	1	1,25/6 = 0,208	1,208
Piraeus	1	-	1
Attica	1	-	1

Greece	2	-	2
Drama ,_Greece	1	1/6=0,166	1,166
Municipality	1	-	1

Selecting nodes. *Splitting to individual graphs.* The Graph possibly consists of independent sub-graphs (Figure 4) that originated from different topics such as music, sports, cars, etc. Our system detects those graphs and treats differently each one, to highlight multi-variety pages relevant to User's interests.

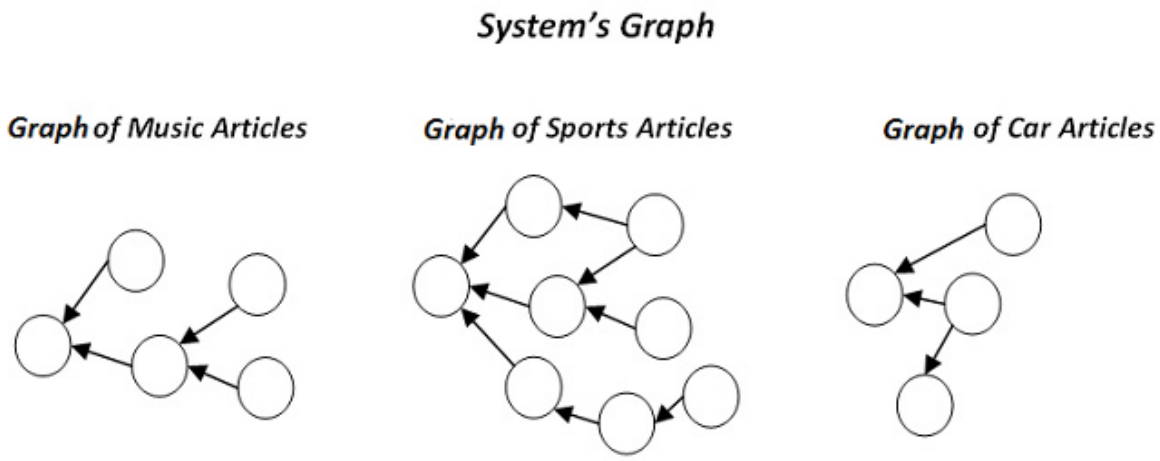


Figure 4: Multiple existence of Graphs

Candidate Nodes per individual graph. We keep only the nodes from each graph that have height 1 or 2. The Articles of these nodes usually contain more useful information. Since, nodes with height 3 or higher tend to be general and not suitable for building profiles. In our example, (Figure 2) nodes "London", "England", "United Kingdom", "Medieval" are valid for recommendation, whereas nodes "Country" and "European History" are not.

Normalize nodes of individual graph and choose nodes to recommend. In Figure 5 we describe the steps we follow to decide which nodes will be recommended to the user. Each graph (Figure 5, L1) is transformed to an array (Figure 5, L2), where each line represents a node from that graph. Each array's nodes are also sorted by descending order count.

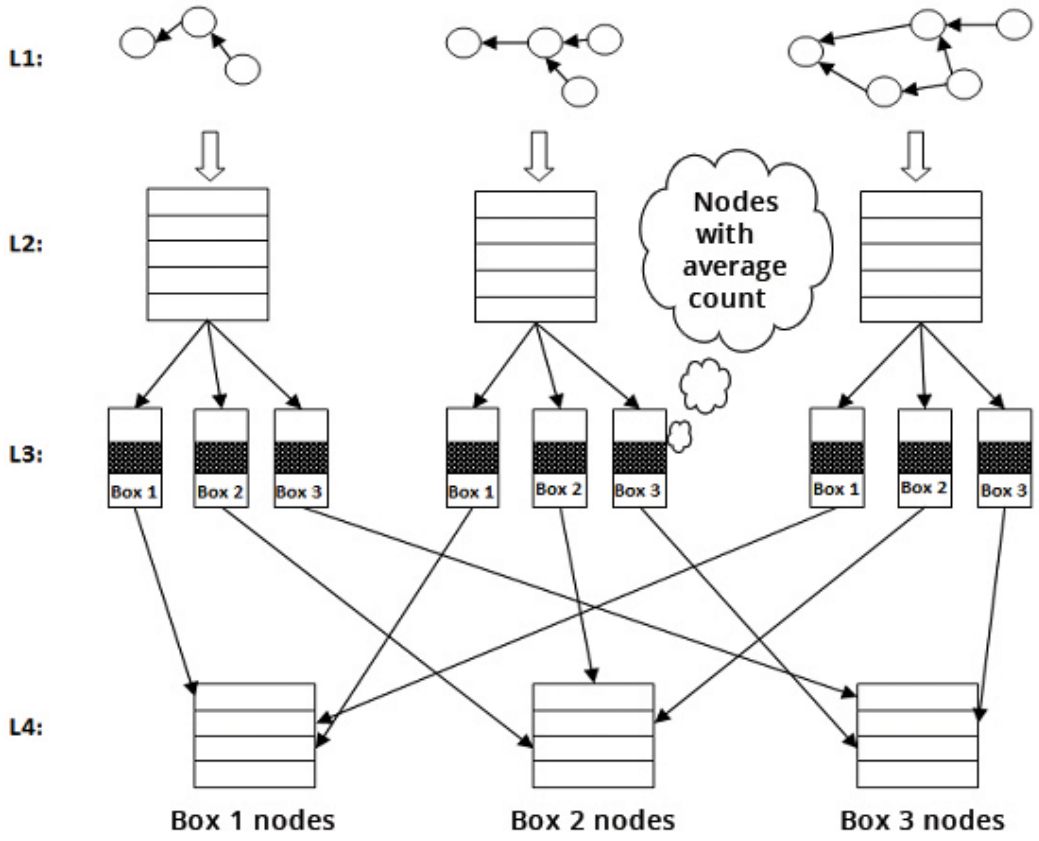


Figure 5: Procedure for selecting Wikipedia articles to recommend

Then graph's nodes are split into 3 "Boxes", depending on their Count (Figure 5, L3). The 1st box contains the graph's nodes that have the largest count, the 2nd box has nodes with middle count and the 3rd one the rest of the nodes. Figure 6 shows a detailed example of how the nodes of a specific graph are split into categories. Now, we select the Rec_Num nodes that we will recommend to the user. The number of results for each graph is relative to its total Count and the total number of results that we desire for each normalization box (Figure 5, L4). So, the 1st box at L4 contains a selection of nodes from all the "Boxes 1" of L3, the 2nd a selection of nodes from all the "Boxes 2" and the 3rd one the nodes from the "Boxes 3". The nodes of all boxes at L3 of a graph that are recommended, are chosen equally on each side of the average count of the current set nodes.

At this point, we will refer to some properties of each Box from this normalization process. "Boxes 1" nodes are general, but selecting nodes with average count from that box, provides useful general information to the user. For example, if the

user has Shared a link from the band "Guns N' Roses", the article "Rock" would be an ideal node for "Boxes 1". On the other hand, nodes of "Boxes 3" are very specific concepts and tightly connected to the actual shares and thus are obvious to the user. However, nodes in the average count of that box contain interesting nodes for "Like" recommendations. For instance, the article "Don't Cry" (Guns N' Roses song) would be a possible node of "Boxes 3". Finally, "Boxes 2" contain nodes with the highest serendipity. More specifically, although these nodes are neither close to the Share's actual context, neither very general, they may provide useful information to the user and help him to discover and learn new things. In the last case, the article "Led Zeppelin" represents an example of a node that "Boxes 2" contain.

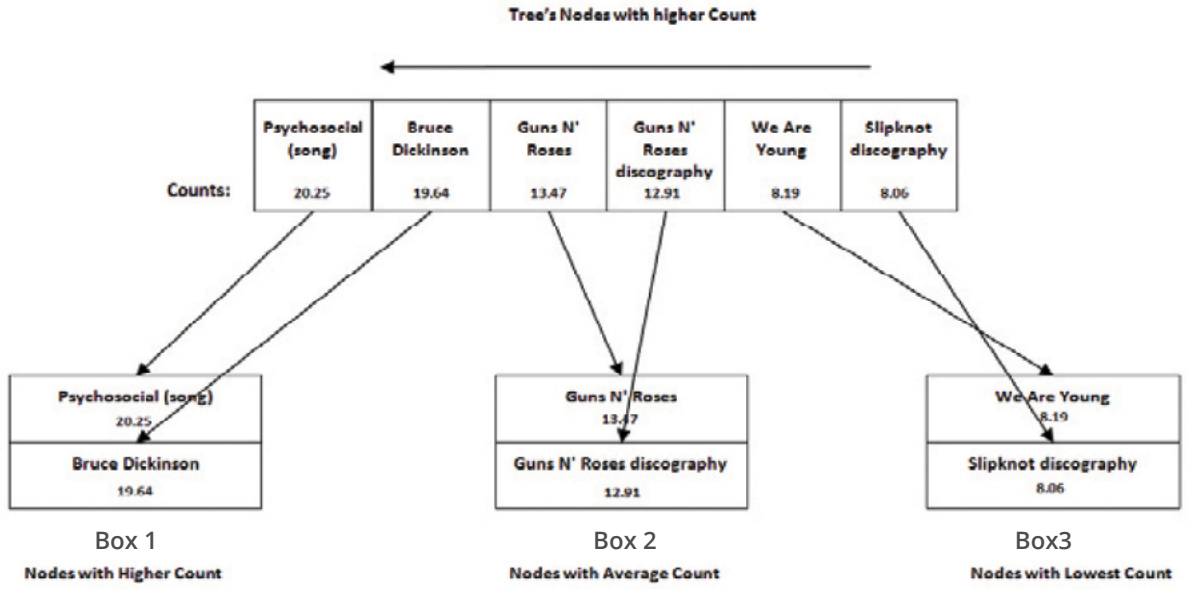


Figure 6: Normalization Method of Graph's Nodes

3. Experiments

3.1. System's Configuration

Table 2 describes the values of the parameters, that we gave to the system during the evaluation. Table 3 describes the values of the parameters we used at the LDA.

Table 2: System's parameters Values

Variable Name	Description	Value		
YT_Count	Youtube Shares Number	30		
K_NumFriends	Friends Number	5		
YT_LinkComments	Comments number	500		
YT_NumRelated-Videos	Related Videos number	5		
Res_GoogleLinks	Google search results number	3		
Wiki_NumLinks	Article's links number	10		
w	Derived Count weight	6		
Rec_Num	Recommendations number per Box/Total	**Total:** 25		
		Box1: 8	**Box2:** 11	**Box3:**6

Table 3: LDA parameters Values

Alpha	beta	Topics number	Iterations number
50/K = number of topics = number of Youtube links	0.1	Number of Youtube links = 30	2000

3.2. Online User Study

Evaluation Preparation. In order to test the accuracy and effectiveness of our System, we created an online Survey where participated 19 people (users). We asked each user to fill a questionnaire with 2 questions, for each recommendation. The structure of the questionnaire is shown at Figure 7.

In the 1st Question the answers "I like it a lot" or "I like it" mean that the user would "Like" a page on Facebook relevant to the topic of the article. If the user chose the option "I don't like it" or "I don't like it at all", not only he wouldn't "Like" a page relevant to the article, but he also didn't like the general topic of the article.

In case the user selected as response the “Neutral” at Question 1, we asked him the extra question “Was the content of the article useful?” with available answers (a) Yes or (b) No. Answer “Yes” means that the user found the context of the article interesting, although he neither Liked it nor Disliked it. Answer “No” means that this article was uninteresting for the user. We asked this extra question because there are many cases where the context of an article is neutral by its definition, but its information might be interesting. For example the article “Crime in the United States”[3] is generally an informative article and is not a subject for Like or Dislike. However, someone who lives or travels in the U.S. may find this information useful.

In the 2nd Question, the option “its Title” means that the user understood the topic of the article by reading only its title. The option “The short Text” means that the user didn’t understand the topic of the article by reading only its title and he continued his reading to the short text. The User chose the option “The webpage of the link”, if he hadn’t still understood the topic of the article from the evaluation page and he read the webpage of the article to read more information about it. In case the user had to browse the internet to find more information about the topic of the article, he selected the option “Browsing on the Internet for more information”.

In our study, we define the Serendipity when a user chose the answer “I like it a lot” or “I like it” at Question1 and selected any answer at Question2 except “Its Title”. In other words, the user liked an article either for its information either for “Like” a relevant Page, after learning new information, that he wouldn’t know before.

Finally, we asked the participants whether they would find useful such a service on Facebook.

3 *http://en.wikipedia.org/wiki/Crime_in_the_United_States*

Figure 7: User Evaluation - Recommendation Page

Results. Tables [4,5] show the main results from the evaluation. In our study, we considered that a recommendation is positive and helpful [Table 4, Like], if the user has selected at Question1 the answer "I like it a lot" (+2), "I like it" (+1) or if he has chosen the "Neutral" (0) response, but also answered "Yes" to the extra question. If the user has selected at Question1 the answer "I don't like it" (-1) or "I don't like it at all" (-2), we suppose the user found this recommendation negative [Table 4, Dislike]. The user found neutral an article [Table 4, Neutral], if he had answered "Neutral" (0) at Question 1 and also answered "No" to the extra question.

Serendipity [Table 5], is measured by the probability P(short_text+web_page+extra_search|likes+interest) and represents the percentage of how often a user

liked an article, whenever he felt had discovered or learnt new information. Finally, the σ represents the Standard Deviation(SD) in our measurements.

Table 4: System's Usage Statistics

	Like	Dislike	Neutral
User Average	73%	12%	15%
σ	14%	10%	10%

I find such a service useful	
Positive	84% (16/19)
Negative	16% (3/19)

Table 5: System's Serendipity Statistics

Serendipity	Title	Short Text + Webpage + Extra Search
User Average	57%	43%
σ	20%	20%

4. Conclusion

In this paper we propose a method for extracting user's interests from Facebook Youtube shares. This method exploits textual features and creates a graph based on Wikipedia's ontology. The recommendations were evaluated in a user experiment and the final results were promising. However, the most important thing is that almost all users would find this service on Facebook useful. Furthermore, we plan to expand our methods beyond English by using international Wikipedias. We plan to improve our results by applying minimum cut in our graph. In this way, we can discover graphs with different topics, that not need to be independent graphs. We will also test other ontologies to take advantage of the provided semantic links.

References

[1] Sandra Garcia Esparza, Michael P. O'Mahony, Barry Smyth: CatStream: categorizing tweets for user profiling and stream filtering In: IUI Santa Monica, CA, USA (2013) 25-36

[2] Matthew Michelson, Sofus A. Macskassy: Discovering users' topics of interest on twitter: a first look. In: AND Toronto, Ontario, Canada (2010) 73-80

[3] Katja Filippova, Keith B. Hall: Improved video categorization from text metadata and user comments. In: SIGIR Beijing, China (2011) 835-842

[4] Xiaoguang Qi, Brian D. Davison: Web page classification: Features and algorithms. In: ACM Computing Surveys (CSUR) Volume 41(2) (2009)

[5] Xing Wei, W. Bruce Croft: LDA-based document models for ad-hoc retrieval. In: SIGIR Seattle, WA, USA (2006) 178-185

[6] David M. Blei, Andrew Y. Ng, Michael I. Jordan: Latent Dirichlet Allocation. In: Journal of Machine Learning Research Volume 3 (2003) 993-1022

ΔΙΠΛΩΜΑΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Methods of Computational Intelligence for Classifying Masses from Mammographic Images

Pavlos P. Kafouris (pavloskaf@di.uoa.gr)

Abstract

Worldwide, breast cancer is among the most common forms of cancer and its incidences come first among the women population. While mammography is regarded as the most reliable technique for screening and diagnosing breast cancer, the interpretation of mammograms is a difficult and error-prone task. In this thesis, we developed methodologies that can be integrated in a computer aided diagnosis system for mammography, concerning the detection and the evaluation of mammographic masses. After developing all subunits of the system, we proceeded on its evaluation in order to investigate its efficacy and its potential to be adopted in daily clinical practice. The optimum settings for classifiers were determined, which were used for the final stage of classification in an independent set.

Keywords: Computational Intelligence, Risk Estimation, Mammographic Imaging, Mass Classification

Advisor

Spyrou George, Staff Research Scientist (Prof. Level)

1. Introduction

Early detection is the key to the ultimate survival rate for breast cancer patients. For women whose tumors were discovered early, the five year survival rate was about 82%, as opposed 60% for the cases not been found early [1]. Among different noninvasive ways for breast cancer diagnosis, mammography is regarded as the most effective tool available today [2]. The characterization of lesions as benign or malignant based on their appearance in mammograms is a difficult task even for expert radiologists. Because a mammogram is a two dimensional projection of a three-dimensional object, superposition of breast tissue often produces patterns that appear like suspicious masses to a radiologist or alters the appearance of real mammographic lesions. This leads to 10%–30% of all cancers to be missed by radiologists [3].

Therefore, computer-aided diagnosis systems (CADx) have been proposed in the past years with the aim to support radiologists in the discrimination of benign and malignant mammographic lesions and to increase the positive predictive value (PPV) of mammogram interpretation. These algorithms are based on extracting image features from regions of interest (ROI) and classifying them accordingly.

In this thesis, a total of 901 mammograms (447 benign and 454 malignant masses) were used, coming from DDSM as in study [4], [5] and [6]. We have developed methods for automatic classification of mammographic masses, based on the basic principles governing a CADx: segmentation, feature extraction, feature selection and classification. The segmentation of the masses from the surrounding tissue was done using the region growing technique. For the description of the masses, we extracted 195 features that were associated with their morphology and texture. In these features, the age of the patient was added. To find the optimal subset of these features, we used the sequential forward selection (SFS) method. Regarding the classification methods, classifiers: k-NN, PNN and SVM were selected. The optimal parameters of these classifiers were determined using the LOO method, while the evaluation of classifiers was performed by the External Cross Validation (ECV) method repeated by 50 times. The results showed that the SVM classifier has achieved the highest separation performance to unknown data, obtaining AUC and an accuracy equal to 0.88. We use one of the larger samples (901 mammograms from DDSM), since study [5] uses 1076 masses from DDSM, which is the largest number, followed by study [7], which uses 427 mammograms

that were randomly selected from the Department of Radiology at the University of Michigan. Study [5] can be considered to be the closest relatively to ours in the sense that the number of mammograms is very large in both, coming from the same database and using the same type of features for the description of the masses. However, in [5] all neural networks were trained using the LOO method and the reported accuracy was 0.81 for image-based and 0.83 for case-based evaluation, while in ours the highest accuracy was 0.88 and the classifiers were evaluated with a more objective method (ECV, 70% training set, 30% testing set). In the other two studies [4] and [6] the number of mammograms obtained from DDSM is smaller, 343 and 236 masses respectively, and they extracted a small number of textural features (7 in [4], 28 in [6]) and morphology (7 in [6]), but still, they achieved lower AUC value (0.86) from our study. Overall, the studies reported to have the highest performance were [8], [9] and [10] in which the AUC got value 0.94, 0.95 and 0.94 respectively. However in these studies, the LOO method was applied to train and test their classifiers and the number of mammograms used was significant smaller than ours (168, 111 and 95 mammograms respectively).

2. Materials and Methods

2.1. DataSet Description

In this thesis we used cases from the publicly available Digital Database for Screening Mammography (DDSM). All volumes of cases have been digitized with: a Lumisys 2000 laser scanner with 12 bit depth and 50 μm pixel size, a Howtek MultiRad 850 with 12 bit depth and 43.5 μm pixel size, a Howtek 960 with 12 bit depth and 43.5 μm pixel size and a DBA (M2100 ImageClear) with 16 bit depth and 42 μm pixel size. We used a set consisting of totally 901 ROIs including 447 benign and 454 malignant masses. Most mammograms had medio lateral oblique (MLO) and craniocaudal (CC) views, but in some cases the mass was only visible in one projection. 70% of the images were used to train the classifiers and 30% for testing them.

2.2. Preprocessing and Segmentation

In this study, we initially performed a preprocessing step through Contrast

Limited Adaptive Histogram Equalization (CLAHE) and afterwards we applied a semiautomatic region-growing approach, with the position of the seed point to be user-defined. The region is iteratively grown by comparing all unallocated neighboring pixels to the region. The difference between a pixel's intensity value and the seed pixel's intensity is used as a measure of similarity. The pixel with the smallest difference is allocated to the respective region. This process stops when the intensity difference between seed pixel and new pixel becomes larger than a certain threshold t=0.1.

2.3. Feature Extraction

Most approaches to feature extraction for mammographic masses are based on the lesion attributes that are used for lesion characterization by radiologists. Radiologists characterize masses based on their shape, the characteristics of their margin, and their optical density. A broad range of techniques for the extraction of features that resemble lesion attributes used by radiologists has been proposed. However, higher-order features that do not directly resemble attributes used by radiologists were also employed [3]. Thus, at this stage we extracted 195 feautures that were associated with the morphology and texture of a mass. Also, in these features, we added the age of the patient.

1) Shape

Based on a segmentation of the mass contour, we proposed several basic morphological features to represent the shape of a mass. These include the area, the perimeter, as well as the circularity, rectangularity, compactness, orientation, major-minor axis and eccentricity of the mass. Furthermore, we used normalized central moments and moments of the border pixels of a mass to represent its shape. Also, we introduced a set of features based on the normalized radial length (NRL) to represent the mass's shape. The NRL is defined as the Euclidean distance of each pixel on the object's contour to the object's centroid.

2) Margin characteristics

We used two features to describe the margin characteristics of a mass which have been proposed by [11], the standard deviation and the skew of the gradient

strength of the pixels on the contour of a mass.

3) Optical density

The optical density of a mass is represented by simple features such as the mean gray level, standard deviation of gray levels, maximal-minimal-median gray level and coefficient of variation of the mass region.

4) Texture

Based on the segmentation of the mass from the background, texture analysis is often restricted to the mass region, excluding the background tissue region. In this study, we extracted 1st order statistics like skewness, kurtosis, energy and entropy gray level, as well as higher-order features that do not represent lesion attributes used by radiologists. Specifically, features based on the gray level co-occurrence matrix (GLCM), gray level run length metrics (GLRLM) and wavelet decompositions were proposed for the characterization of masses.

2.4. Feature Selection

In this study, we used a wrapper approach to find the optimal feature subset, because the main advantage of this method is the interaction with the classifier and the correlation between the different features. However, its disadvantage is the increase in complexity and time in order to find the desired subset of features. Specifically, we used the Sequential Forward Selection (SFS) approach [12], which is a sequential feature selection method. The SFS approach, finds initially the best single feature in terms of maximum accuracy for the separation of the two classes. Then, that feature is adopted permanently and put in combination with each of the other features in turn. Out of all the pairs created, the best set of two features is taken, again in terms of the maximum accuracy for the separation in the feature space. The process continues until all features are selected.

2.5. Classification

The discrimination of benign and malignant mammographic masses is a super-

vised learning problem, which is defined as the prediction of the value of a function for any valid input after training a learner using examples of input and target output pairs. For the problem at hand, the function has only two discrete values, hence the problem of discriminating benign and malignant masses can be modeled as a two-class classification problem [3].

1) Classifiers

The classifiers that have been applied to solve this problem are: k - nearest neighbors (k-NN), probabilistic neural network (PNN) and support vector machines (SVM).

The k-NN classifier is based on the calculation of the similarity between a vector to be classified and the vectors of the training set. In this study, the Euclidean distance was used as a measure of similarity between two vectors.

For the PNN classifier we used the following discriminant functions [13], [14]:

$$g_j(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}\sigma^p N_j} \sum_{i=1}^{Nj} e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}} \quad \text{(Gaussian)} \quad (1)$$

$$g_j(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}\sigma^p N_j} \sum_{i=1}^{Nj} e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|}{\sigma}} \quad \text{(Exponential)} \quad (2)$$

$$g_j(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}\sigma^p N_j} \sum_{i=1}^{Nj} \frac{1}{1+\|\mathbf{x}-\mathbf{x}_i\|^2/\sigma^2} \quad \text{(Reciprocal)} \quad (3)$$

where x is the test pattern vector to be classified, x_i is the i-th training pattern vector, N_j is the number of patterns in class j, σ is a smoothing parameter, and p is the number of features employed in the feature vector. The test pattern is classified to the class with the larger discriminant function value.

In the SVM classifier, the separating hyperplanes in the transformed feature space

are defined by equations (4):

$$w\times\ \Phi(\mathrm{x})+\mathrm{b}=\pm 1 \tag{4}$$

where +1 is referred to class 1, –1 is referred to class 2, x is the pattern vector, w is a vector perpendicular to the hyperplanes, and b the bias or threshold which describes the distance of the decision hyperplane from the origin (equal to $b/||w||$). The discriminant function is given by:

$$g(x)=sign(w\times\ \Phi(\mathrm{x})+\mathrm{b}) \tag{5}$$

Functions that were used as kernels $\Phi(x)$ are:

i) The linear kernel: $k(xi,xj) = xi.xj$ (6)

ii) The polynomial kernel: $k(xi,xj) = (xi.xj +\theta)d$ (7)

where d the order of polynomial and θ an offset parameter.

iii) The Gaussian radial basis kernel:

$$k(x_i,x_j)=\exp\left(\frac{-(x_i-x_j)^T(x_i-x_j)}{2\sigma^2}\right) \tag{8}$$

where σ is the standard deviation.

iv) The sigmoidal kernel: $k(xi,xj) = tanh(\kappa(xi.xj)+\theta)$ (9)

where κ the gain and θ the offset.

2) Parameters

From the previous section it is observed that the classifiers depend on some parameters, whose values influence their performance directly. Therefore, an appropriate parameter selection stage must be preceded for the training of each classifier. However, the best choice of parameters depends on the data; therefore, the optimal parameters may have different values.

For the k-NN classifier, the number of nearest neighbors, k, has to be checked. In binary classification problems, such as ours, it is helpful to choose k to be an odd number as this avoids tied votes. Thus, this parameter took values from

the set {3, 5, 7, 9, 11, 13, 15, 17, 19, 21}, resulting in designing 10 different k-NN classifiers. Regarding the PNN classifier, the σ parameter can affect the classification results. For all three types of the PNN classifier (Gaussian, Exponential, Reciprocal), the search for the optimal value of this parameter was made in the range [0.1-1] by varying the value with step 0.1. For the polynomial SVM classifier, we searched optimal values of the d parameter, i.e. the degree of the polynomial, while for the rbf SVM classifier we searched for optimal values of the σ parameter, i.e. the standard deviation. The possible values of the d parameter were from the set {2, 3, 4, 5}, while the σ parameter got values in the [0.4-1.4] range by varying the value with step 0.2.

So, in fact, we designed 52 variations of 3 types of classifiers (10 k-NN, 30 PNN, 12 SVM), and after their evaluation, we obtained the best classifier from each category.

3) The evaluation of classifiers

The computer classification results were validated using the following standard criteria: Accuracy (ACC), Sensitivity (SN), Specificity (SP) and the area under the ROC curve (AUC).

3. Results

3.1. Training and Classification

For each of the 52 classifiers, the SFS method was applied and using the LOO method, the best combination of features were determined, i.e. the combination which had achieved the highest accuracy. For a given round, the LOO method used one case for testing and the remaining cases for training. The case chosen for testing was changed at each round, until all cases had been used once for testing. For the k-NN and PNN (Gaussian, Exponential, Reciprocal) classifiers we observed that the maximum accuracy value (>0.83) was achieved in the first 10-13 features approximately, and as we increased the number of features used it declined. Different results were obtained for the SVM classifiers. For the

polynomial SVMs and rbf SVMs classifiers, the maximum accuracy equal to one was achieved in the first 8-10 features approximately and as we increased the number of features their accuracy remained stable. In the SVM sigmoid classifier the maximum accuracy value (0.89) was achieved after 102 features, and in the SVM linear classifier the maximum accuracy value (0.67) was achieved after 135 features, which was the smallest maximum accuracy value compared to all the other classifiers. As we increased the number of features used, the accuracy of both classifiers declined.

3.2. Evaluation

The evaluation of classifiers was performed by the ECV method, which is based on recurrent training and evaluating of a classifier using different subsets of cases. All the cases were randomly divided into two subsets in the 70%-30% ratio: the first one was used for the training of classifiers and the second one for the evaluation. This process was repeated by 50 times. The optimal parameters of the classifiers were determined based on the combination of the mean value and the range of accuracy, which were calculated after the 50 repeated uses of the classifiers. The results showed that the optimal value of the k parameter of the k-NN classifier is equal to 13, because the 13-NN classifier, after 50 repeated uses, gave the highest mean value of accuracy (0.81) with one of the smallest ranges of accuracy (0.11). For the PNN Gaussian classifier the optimal value of the σ parameter is equal to 0.1, since after 50 repeated uses of the classifier, the highest mean value of accuracy (0.83) was achieved with the smallest range of accuracy (0.08). For the PNN Exponential classifier the optimal value of σ is equal to 0.4, while the PNN Reciprocal's is 0.1. In the SVM classifiers, the SVM polynomial classifiers outweigh the others, since they achieved the highest mean value of accuracy ranging from 0.73 to 0.88 with a small range of accuracy, while the best SVM rbf classifier achieved a highest mean value of accuracy equal to 0.7, the SVM linear classifier achieved an accuracy of 0.42 and the SVM sigmoid classifier an accuracy of 0.49 with greater range. From the polynomial SVM classifiers the 2nd degree SVM polynomial gave the highest mean value of accuracy 0.88 with the smallest range of accuracy (0.09).

3.3. Classification in an independent set

In this section, we present the results of the final classification in unknown data. For the comparison of the classifiers, we used again the ECV method, in which, for all classifiers, the training set (70%) consisted of the data by which they were constructed (313 cases of benign and 318 cases of malignant), while the test set (30%) contained 134 new cases of benign and 136 new cases of malignant, in order to determine the performance of the classifiers using completely unknown data. Table 1 shows the corresponding values for each classifier and Figure 1 illustrates the ROC curves of the best classifiers of each class in a common graph.

Therefore, the best performance for the discrimination of mammographic masses was achieved by the SVM polynomial classifier with degree=2 and N=25 (length of the best features combination). The PNN Gaussian classifier with σ=0.1 and N=10 had the second best performance, and then the rest PNN classifiers follow with similar behavior.

Table 1:
MEASURES OF PERFORMANCE OF CLASSIFIERS USING ECV METHOD

Classifier	Acc	Sn	Sp	AUC
k-NN (k=13, N=12)	**0.804**	**0.765**	**0.843**	**0.804**
PNN Gaussian (σ=0.1, N=10)	**0.867**	**0.875**	**0.858**	**0.867**
PNN Exponential (σ=0.4, N=13)	0.844	0.838	0.851	0.845
PNN Reciprocal (σ=0.1, N=14)	0.837	0.831	0.843	0.837
SVM Linear (N=135)	0.396	0.427	0.366	0.396
SVM Polynomial (degree=2, N=25)	**0.878**	**0.882**	**0.873**	**0.878**
SVM RBF (σ=0.6, N=8)	0.637	0.566	0.709	0.638
SVM Sigmoid (N=102)	0.600	0.596	0.605	0.600

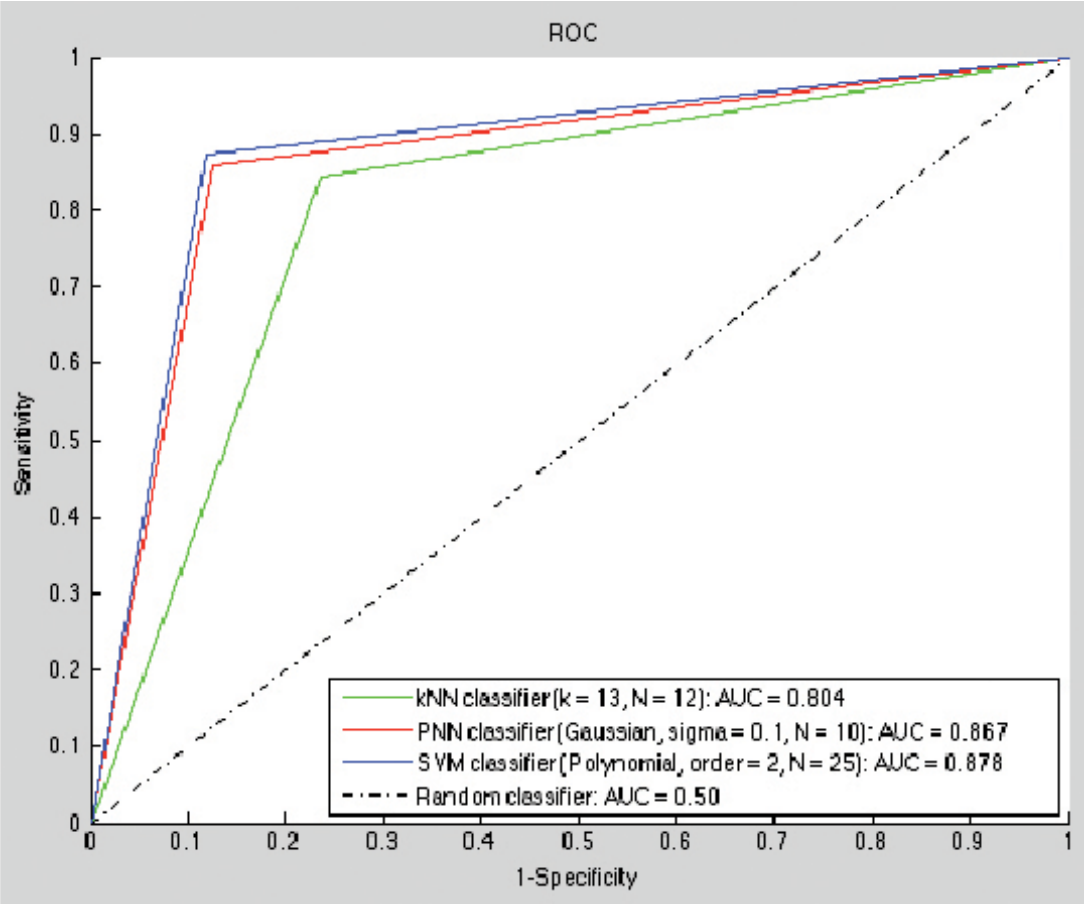


Figure 1: The ROC curves of the classifiers

4. Conclusions

In this thesis, we developed methods for automatic classification of mammographic masses, based on the basic principles governing a CADx. The region growing technique was used for the segmentation of the masses from the surrounding tissue, starting from a seeding point, which is selected by the user. For the description of the masses, we extracted 195 features that were associated with their shape, the characteristics of their margin, their optical density and their texture. Furthermore, in these features we added the age of the patient. SFS was used to find the optimal feature subset, thus increasing the performance of classification and ensuring simultaneous reduction of the complexity of the problem. Regarding the classification methods, k-NN, PNN and SVM classifiers were selected. A total of 901 mammograms (447 benign and 454 malignant

masses) were used, coming from the publicly available DDSM. Applying the SFS and using the LOO method, the best combination of features and the optimal parameters of classifiers were determined, while the evaluation of classifiers was performed by the ECV method repeated by 50 times. All the cases were randomly divided into two subsets in the 70%-30% ratio: the first one was used for the training of classifiers and the second one for the evaluation. After the calculation of the performance of each classifier in the evaluated set, the best of k-NN, PNN and SVM classifiers were determined, which were used for the final stage of classification in an independent set. More specifically, the 13-Nearest Neighbors classifier with 12 from total 196 features (k-NN, k=13, N=12) achieved an AUC value = 0.80, the Gaussian PNN classifier with standard deviation equal to 0.1 and N=10 (Gaussian PNN, σ=0.1, N=10) got AUC value 0.87 and the polynomial SVM classifier with degree=2 and N=25 (Polynomial SVM, d=2, N=25) achieved an AUC value = 0.88. This large scale analysis under multiparametric optimization questions gives useful information about the optimum selection for a classification scheme regarding mammographic masses. These results are expected to enlighten the design of Computer Aided Diagnosis systems focusing on mammographic mass classification.

References

[1] J. Ferlay, H. R. Shin, F. Bray, D. Forman, C. Mathers, and D. M Parkin. Globocan 2008 v1.2. cancer incidence, mortality and prevalence worldwide in 2008. IARC CancerBase No. 10. Lyon, France: International Agency for Research on Cancer; 2010. [Accessed December 1, 2011]. at http://globocan.iarc.fr/, 2008.

[2] J. Tang, R. Rangayyan, J. Xu, I. Naqa, and Y. Yang, «Computer-aided detection and diagnosis of breast cancer with mammography: Recent advances,» IEEE Transactions on Information Technology in Biomedicine, vol. 13, pp. 236-251, 2009.

[3] Elter, M. and Horsh, A. 2009. A CADx of mammographic masses and clustered microcalcifications: a review, Med. Phys., 36, 2052–2068.

[4] Lim WK, Er MJ. Classification of mammographic masses using generalized

dynamic fuzzy neural networks. Med Phys. 2004 May; 31(5): 1288-95.

[5] C. Varela, S. Timp, and N. Karssemeijer. Use of border information in the classification of mammographic masses. Phys Med Biol. 2006 Jan 21; 51(2):425-41. Epub 2006 Jan 4.

[6] J Liu, J Chen, X Liu, and J. Tang. An investigate of mass diagnosis in mammogram with random forest. In Advanced Computational Intelligence (IWACI), pages 638 – 641, 2011.

[7] Jiazheng Shi, Berkman Sahiner, Heang-Ping Chan, Jun Ge, Lubomir Hadjiiski, Mark A. Helvie, Alexis Nees, Yi-Ta Wu, Jun Wei, Chuan Zhou, Yiheng Zhang, and Jing Cui. Characterization of Mammographic Masses Based on Level Set Segmentation with New Image Features and Patient Information. Med Phys. Jan 2008; 35(1): 280–290.

[8] Sahiner B, Chan HP, Petrick N, Helvie MA, Goodsitt MM. Computerized characterization of masses on mammograms: the rubber band straightening transform and texture analysis. Med Phys. 1998 Apr; 25(4): 516-26.

[9] Mu T, Nandi AK, Rangayyan RM. Classification of breast masses using selected shape, edge-sharpness, and texture features with linear and kernel-based classifiers. J Digit Imaging. 2008 Jun;21(2):153-69. doi: 10.1007/s10278-007-9102-z. Epub 2008 Feb 28.

[10] Z. Huo, M. L. Giger, C. J. Vyborny, D. E. Wolverton, R. A. Schmidt, and K. Doi. Automated computerized classification of malignant and benign masses on digitized mammograms. Acad Radiol. 1998 Mar; 5(3): 155-68.

[11] Timp S., Karssemeijer N. A new 2D-segmentation method based on dynamic programming applied to computer aided detection in mammography. Med. Phys. 31 958–71, 2004.

[12] Theodoridis S, Koutroumbas K (2005) Pattern recognition. Academic, New York.

[13] Specht D, Probabilistic Neural Network, Neural Networks, Vol. 3, pp. 109-118, 1990, page 118, Lecture Notes.

[14] M. Hajmeer, I. Basheer, A probabilistic neural network approach for modeling and classification of bacterial growth/no-growth data, Journal of Microbiological Methods 51, 217– 226, (2002), page 128, Lecture Notes.

Robbing, Surfing and Rioting Games on Graphs

Ioannis Lamprou (ilamprou@di.uoa.gr)

Abstract

The focus is on 3 different combinatorial pursuit-evasion games. For the famous Cops & Robber game, where cop tokens try to capture a robber token on a graph, two variants are examined. That is, the cases of fractional cops and fast robber. Grid graphs are mostly studied, besides some general results. For the recently introduced Surveillance game modeling Web prefetching, a hardness result for bounded degree graphs is provided. Here, a marker tries to anticipate any possible itinerary of a surfer. Finally, Eternal Domination is studied. Guards need to perpetually dominate a graph and thus protect against an attack on any node.

Keywords: Cops & Robber, Surveillance, Eternal Domination, 2-Player Games, Pursuit-Evasion Games

Advisor

Vassilis Zissimopoulos, Professor

1. Introduction

Historically, games have always appeared in human societies and in many different forms. In recent years, mathematical studying of games has received great interest by researchers in many fields in an attempt to formalize them. We concentrate on a particular subset of Game Theory, namely Combinatorial Game Theory, which includes games with specific characteristics. The interest is about games where two players play alternately and both enjoy perfect information over the game. That is, at each turn, each player is familiar with the current and all previous game configurations and picks an action out of a set of publicly known predefined actions. Moreover, the games are deterministic in the sense that no action performed is dependent on any source of randomness. By the end of the game, one player wins and the other loses; there may be no tie.

Here, we focus on three pursuit-evasion combinatorial games played on graphs. Consequently, we identify a strong interaction with Graph Theory for these games. In other words, the features of the graphs, on which the games are described and played, have a major impact on the dynamics emerging between the two players. The games under consideration are the Cops & Robber game, the Surveillance game and the Eternal Domination game.

1.1. Cops & Robber

Cops & Robber is a pursuit-evasion combinatorial game played on a graph. There are two players: one that controls the cop tokens (player C) and another who controls the robber token (player R). Initially, C places his k tokens on the vertices of the graph. Notice that more than one cop tokens may lie on the same node. Then, R chooses an initial placement for the robber. Round 0 is over. From now on, every round consists of 2 turns, where C may or may not move any of his cops to a vertex adjacent to the one he currently lies on and R moves the robber to an adjacent vertex with respect to her current position or does not move her at all. C wins the game if, after any player's turn, the robber lies on the same vertex with a cop. R wins if she can perpetually avoid the realization of the aforementioned condition.

1.1.1. Background

From the optimization point of view, the important question in mind is what is the minimum number of cops needed to capture the robber either on a specific graph or in general.

Definition 1. The cop number of a graph G, denoted $cn(G)$, is the minimum number of cops needed to ensure that the robber is captured, regardless of her strategy.

Problems related to the cop number have been studied heavily over the last 30 years. Originally, Quillot [23] and Nowakowski and Winkler [22] characterized graphs with cop number equal to 1. Aigner and Fromme [1] proved that $cn(G) \leq 3$ for any planar graph G. Frankl [8] proved a lower bound for graphs of large girth. Moving onto general graphs, Meyniel conjectured that $\sqrt{n}$ cops are always sufficient to capture the robber. Chiniforooshan [5] proved an $O(n/logn)$ upper bound, which was improved by Scott and Sudakov [25] and Lu and Peng [21] to $O(n2^{(1-o(1))\sqrt{\log n}})$.

Computationally, we are interested in the following question: given a graph G and an integer k, does $cn(G) \leq k$ hold? Goldstein and Reingold [15] proved that the problem is EXPTIME-complete given that the graph is directed or the initial positions are given. Recently, Kinnersley [17] answered the question definitely by proving EXPTIME-completeness.

Cops & Robber can be encountered in a variety of fields, e.g. in robot motion planning [26], routing [19] and network security [3].

1.1.2. New Remarks

We examine two variants of the original game and provide several combinatorial results, especially regarding the cop number of small square grid graphs.

1.2. Surveillance

The Surveillance game is a combinatorial game played on a (directed) graph G. There are two players: the marker (otherwise called observer) and the surfer (otherwise called fugitive), who take turns alternately. In trivial round 0, the marker

marks a predefined vertex v_o and the surfer places herself on v_o. Then, for all following rounds, the marker marks a selection of at most $k \in \mathbb{N}$ unmarked nodes, while the surfer chooses a neighboring node to move herself to. Once a node is marked, it remains marked for the rest of the game. The marker wins if he manages to mark all vertices of G before the surfer manages to reach an unmarked vertex, in which case she wins.

Definition 2. The surveillance number of a graph G, with respect to a vertex $v_o \in V(G)$, is the minimum number of marks needed such that the marker wins against any surfer strategy for a Surveillance game starting at v_o and it is denoted $sn(G, v_o)$.

1.2.1. Background

The game was introduced quite recently by Fomin et al. [7] as a model for Web pages' prefetching, where a browser may download potential web pages in advance in order to enhance the user's surfing. They extensively study the computational complexity of the problem of determining $sn(G, v_o)$: they prove PSPACE-completeness and several NP-hardness results. Moreover, they demonstrate polynomial-time algorithms in the case of interval graphs and trees. Finally, the connected variant is considered, where there exists the restriction that, at any round, the subset of nodes marked must be connected. The problem in question is to define the cost of connectivity: how many more marks per round are needed to satisfy the connected variant's restriction [10].

1.2.2. New Remarks

We prove that computing $sn(G, v_o)$ is NP-hard even for directed graphs of maximum degree 6 by providing a complexity reduction from a special case of Vertex Cover. This result is published in [11].

1.3. Eternal Domination

Eternal Domination can be regarded as a combinatorial graph game. There exist two players: one of them controls the guards, while the other controls the rioter. Initially, the guard tokens are placed such that they form a dominating set on G.

Then, the rioter attacks a vertex without any guard on it. A guard, that dominates the attacked vertex, must now move on it to counter the attack ensuring that the modified guard positioning remains dominating. The game proceeds in similar fashion in any subsequent rounds. The guards win this game if they can counter any attack of the rioter and perpetually maintain a dominating set. The rioter wins if she manages to force the guards to reach a positioning that is no longer dominating. Finally, notice that we allow more than one guards lying on the same node.

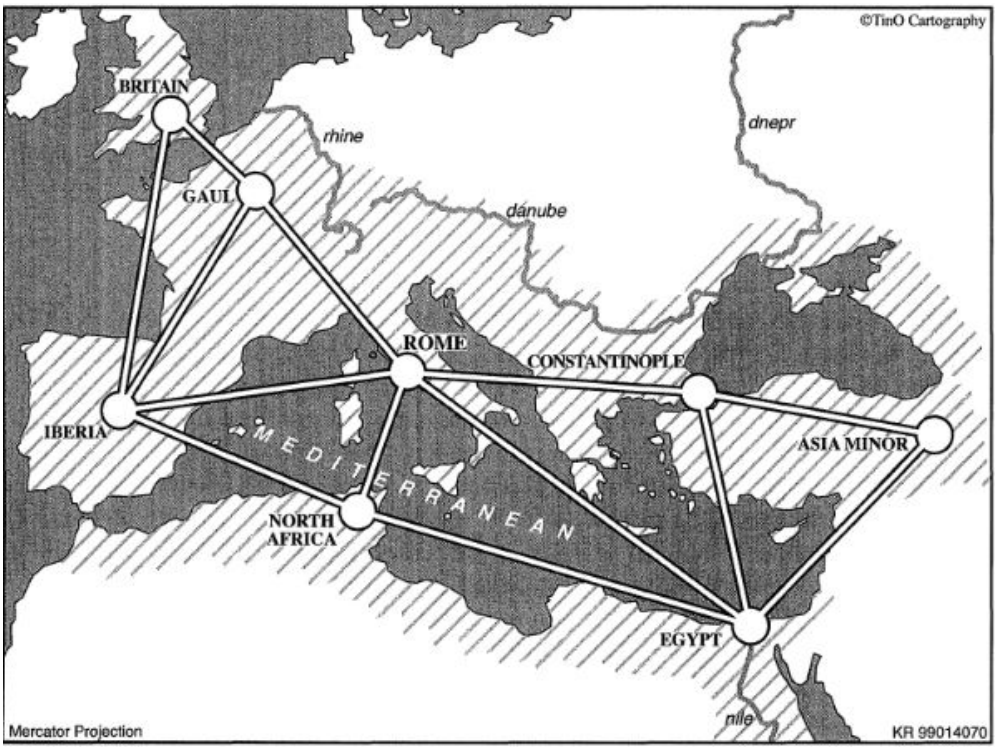


Figure 1: Perpetually protecting the Roman Lake [24]

1.3.1. Background

The idea about infinite order domination was originally considered in [4] as an extension to other domination variants. Later, Goddard et al. [14] focused on the formalisms, which we too follow. More specifically, they consider two variants of the game. In the former, only one guard is allowed to move in each guards' turn, while, in the latter, at most m guards can move in each guards' turn, where $m \in \{2,3,\ldots,n\}$.

Definition 3. The minimum number of guards, needed to perpetually ensure

domination in a graph G against any rioter strategy, is called:

- the eternal 1-domination number of G and is denoted by $\sigma_1(G)$, if only one guard is allowed to move at each round
- the eternal m-domination number of G and is denoted by $\sigma_m(G)$, if at most m guards are allowed to move at each round

Several bounds are obtained in [14]. Many other papers have been published under this setting; see [2, 13, 18].

Applications of such games emerge either on real-life or on computer security situations. Optimizing military defence is a problem that dates back to the Roman and Byzantine empires and relations can be established to networks and autonomous systems [16, 20].

1.3.2. New Remarks

We prove a hardness and an approximation result for two cases of $\sigma_m(G)$.

2. Fractional Cops & Fast Robber

Relaxing the description is a general technique followed in order to augment the understanding on a hard combinatorial problem. Furthermore, the relaxed version could provide an approximation for the original one. Authors of [12] study a natural relaxation for the cop number of a graph, namely the fractional cop number of a graph. The fractional cop number (in short fcn) refers to the original Cops & Robber game, but with the relaxation that the cops can now split into infinitely small and infinitely many pieces and move such pieces along the edges of the graph. The robber remains integral (i.e. she cannot split). Furthermore, in [12] it is proved that splitting would not assist her towards escaping. The sum of all cop pieces remains always equal to a constant $k \in \mathbb{R}^+$. In order for the robber to be captured, a quantity of cops ≥1 needs to lie on the same vertex as her. We now review a cop-strategy and then present some new remarks on it. This game relaxation is referred to as Fractional Cops & Robber.

2.1. A Uniform Strategy

We discuss a variation of the strategy given in [12], which does not substantially differ from the original one, but it helps us simplify the remarks that follow: Initially, the k cops are placed uniformly on the graph, i.e. k/n cops are placed at each node. Then, the robber places herself to a node v. Now, it's the cops' turn. The k/n cops that lie on the same vertex as the robber will (from now on) always follow the robber. The rest $k - k/n = k\ (n-1)/n$ cops are spread uniformly on the graph (we later show that this can be done in exactly 1 cops' step for any graph). That is, each vertex is now guarded by $k\ (n\text{-}1)/n^2$ cops. We do not ever reconsider cop quantities that are bound to always follow the robber. Now, it's the robber's turn. Whatever her move, the cops will repeat the same strategy, i.e. the quantity that lies on the same vertex will always follow her from now and the rest are re-spread uniformly over the graph. Inductively, at step t there will be $x_t = k\left(\frac{n-1}{n}\right)^t$ cops left on the graph, which are not bound to always follow the robber. The rest $y_t = k - k\left(\frac{n-1}{n}\right)^t = k\left(1 - \left(\frac{n-1}{n}\right)^t\right)$ cops accumulate on the robber until this quatity eventually becomes ≥ 1 at a certain round, hence cops win. From now on, we refer to this strategy as the spread and follow strategy.

It is proved [12] that the fractional cop number of any graph approaches 1 if we allow a very large number of steps. One need only notice that $lim_{t\to\infty} x_t = 0$ and so for the quantity that accumulates on the robber $lim_{t\to\infty} y_t = k$. That is, for any $k \geq 1$ the above strategy manages to accumulate all cops on the robber. Obviously, just 1 is enough to capture her, leading us to the following result.

Let $fcn_\infty(G)$ stand for the fractional cop number of graph G for a Fractional Cops & Robber game of infinite duration.

Theorem 1. [12] $\forall G : fcn_\infty(G) = 1$.

The aforementioned result suggests that the fractional cop number does not yield any help towards the approximation of the integral cop number, since it is always (almost) 1. Another suggestion to try to reduce the cops' power, in order to possibly narrow the gap between the fractional and the integral cop numbers, is to bound the number of steps allowed to them (i.e. reduce the duration of the game). Unfortunately, upper-bounding the number of steps allowed in the game still does

not help us in our objective to find a relation to the integral cop number, since the bounded-time fractional cop number remains small enough. The following table summarizes the lower bounds on cop quantity obtained in this scope.

Table 1: Fractional cop numbers for bounded duration games

Steps Allowed	Cops Required
∞	1
n	1.58
$\sqrt{n}$	$\sqrt{n}$
$\log n$	$O(n)$

For the reasoning made to be completely accurate, we need to make sure that the cops' strategy is feasible. We prove the following.

Lemma 1. The k cops have followed the "spread and follow" strategy for $t \geq 1$ rounds of Fractional Cops & Robber on graph G. The robber moves to vertex v during her turn at round t. The total cop quantity lying on $V(G) \setminus \{v\}$ can be moved such that it lies uniformly on $V(G)$ after cops' turn at round $t + 1$.

2.2. Fast Robber

We now turn our attention to the Fractional Cops & Robber game variant where the robber can move with speed 2, i.e. at each step she can move on a path of length at most 2 from her current position. Several natural questions arise in this case about the definition of the game e.g. Can the robber jump over some cop quantity? What happens when a robber co-exists with some cop quantity on a vertex? We try to define the game in a way that it handles such questions, nevertheless it remains as natural as possible. Furthermore, we try to understand whether fractional cops can perform better than integral ones in this context. We focus on square grid graphs and obtain some bounds for the (fractional) cop number on small grids.

Definition 4. Fractional Cops & Fast Robber is the same game as Fractional Cops & Robber with the extension that the robber has speed 2, i.e. she can move from her current position to any vertex at distance at most 2.

Let the k-neighborhood ($k \geq 0$) of a node $v \in V(P_n \times P_n)$ stand for the set of nodes at distance exactly k from v and denote it by $N_k(v)$. In addition, let $N_{[k]}(v) = \bigcup_{0 \leq i \leq k} N_i(v)$.

On the next table, we present some values obtained for the fractional as well as the integral cop number for small $n \times n$ grids ($n \leq 5$). The fractional numbers proven mostly derive from a domination analysis of $N_{[2]}(v)$ for any possible robber position v.

Table 2: Small grids' (fractional) cop numbers for a fast robber

n	$fcn_2(P_n \times P_n)$	$cn_2(P_n \times P_n)$
1	1	1
2	4/3	2
3	2	2
4	2	2
5	$\in [2, 3]$	3

We now introduce a new notion to assist us on the fractional analysis. Let $fdn_{v,[s]}(G)$ stand for the the fractional domination quantity needed to dominate all nodes at distance at most s from v in graph G, but with the extra restriction that the quantity on v is strictly less than 1. We make use of this quantity, since we wish to consider the amount of cops needed to capture the robber after she places herself on vertex v. The extra restriction is put, since there cannot be $a \geq 1$ cop quantity on v, otherwise the game would be immediately over. The linear program below computes $fdn_{v,[s]}(G)$ for any node $v \in V(G)$. In the following figure, let c_i stand for the cop quantity at node i.

$$\begin{array}{ll} Minimize \quad \sum_{i \in V(G)} c_i & such\ that \\ \sum_{j \in N[i]} c_j \geq 1 & \forall i \in N_{[s]}(v) \\ c_v < 1 & \end{array}$$

Figure 2: Linear program for the $fdn_{v,[s]}(G)$ of any node $v \in V(G)$

The following figure provides the 2-neighborhood domination cases in a 5×5 grid.

Notice that a similar partitioning of the vertices can be made for any $n \times n$ grid where $n \geq 5$. The $side_{n \times n,}$ $inner\text{-}side_{n \times n}$ and $core_{n \times n}$ sets are expanded to contain all such (equivalent under positioning) nodes. Using this approach, we comprehend that in a big grid, given strictly less than 4 cops, the robber can move from turn to turn in such a way that she always remains on core nodes.

Corollary 1. For n big enough: $fcn_2 (P_n \times P_n) \geq 4$.

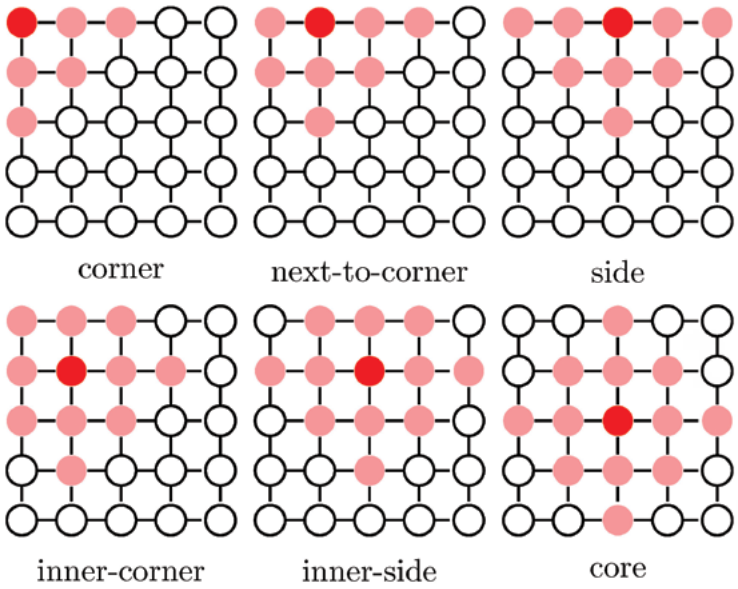


Figure 3: Vertices to be dominated for robber positions on the 5×5 grid

3. Bounded-degree Surfing

An open question in [7] suggested the study of the complexity of Surveillance, when the game is played on graphs whose maximum degree is bounded. Notice that computing $sn(G, v_0)$ is trivial for graphs of maximum degree 3 [7]. The result presented in this section acts as an initial step in estimating the surveillance number complexity for graphs whose degree is bounded by a constant bigger than 3. The following theorem appears in [11].

Theorem 2. Deciding whether $sn(G, v_0) \leq 2$, for a directed acyclic graph G of maximum degree 6 and a starting vertex $v_0 \in V(G)$, is NP-hard.

Below, we may refer to the decision problem in question as the Surveillance Number problem. To prove NP-hardness, a reduction from an NP-hard [9] special case of the well-known Vertex Cover problem is employed.

Definition 5. Vertex Cover for Cubic Graphs: Given a cubic (each node of degree 3) graph G and a constant k, decide whether there exists a set $V'(G) \subseteq V(G)$ such that for any $e = \{v_i, v_j\} \in E(G) : v_i \in V'(G) \vee v_j \in V'(G)$ and $|V'(G)| \leq k$.

From now on, we shall refer to this problem shortly as VC-3. Given an instance (G, k) of VC-3, we transform it into an instance (D, v_0) of the Surveillance Number problem as follows: Let $n = |V(G)|$ and $m = |E(G)|$. For each vertex $v_i \in V(G)$, we put a corresponding vertex $u_i \in V(D)$. That is, n new nodes are added, namely $u_1, u_2, \ldots, u_n$. Then, for each edge $e_i \in E(G)$, we put a corresponding node $c_i \in V(D)$. That is, m new nodes are added, namely $c_1, c_2, \ldots, c_m$. Another $k+m-1$ nodes are added to $V(D)$, namely $v_0, v_1, \ldots, v_{k+m-2}$. As far as the edge set is concerned, $v_0, v_1, \ldots, v_{k+m-2}$, $c_1, c_2, \ldots, c_m$ form a directed path in this order in D. Moreover, each vertex $c_i \in V(D)$ is connected to $u_j, u_k \in V(D)$, where $e_i = \{v_j, v_k\} \in E(G)$, via a specific gadget (see figure). Notice that since G is cubic, each node $u_i \in V(D)$ receives 6 edges, 2 from each corresponding edge's gadget.

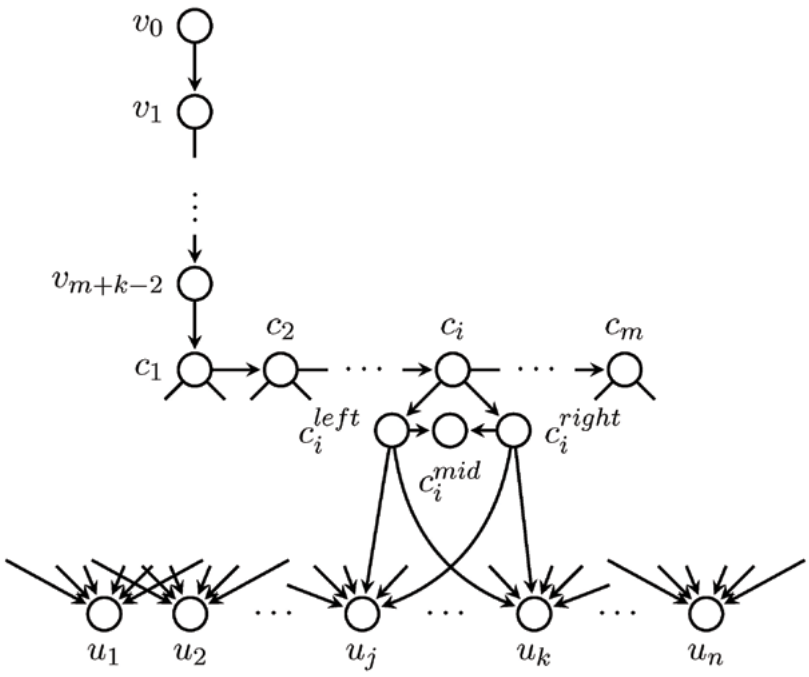


Figure 4: The digraph D constructed for the reduction

We prove the following by providing specific cop-win and robber-win strategies on D using the existence or not of a k-cover for G.

Lemma 2. If G has a vertex cover of size at most k, then $sn(D, v_0) \leq 2$.

Lemma 3. If G has a vertex cover of size greater than k, then $sn(D, v_0) > 2$.

4. The Complexity of Eternal Security

We consider the computational complexity of the following decision problem: Given a graph G and a positive integer k as input, decide whether $\sigma_{1(m)}(G) \leq k$. The minimal complexity class known to include the problem is EXPTIME.

To prove NP-hardness for computing $\sigma_m(G)$, we provide a reduction from the NP-hard MINIMUM DOMINATING SET problem: Given a general graph G and a positive integer k, decide whether there exists a dominating set of size at most k in G.

Given a graph G, we construct a new graph G' as follows: for any vertex $v \in V(G)$, v remains in $V(G')$ and a true twin v' is added as well. Then another vertex, namely u, is added and all edges $\{u,v'\}$ are added. That concludes the construction.

We prove the following two lemmata which we then combine to prove the NP-hardness theorem.

Lemma 4. If G has a dominating set of size at most k, then $\sigma_4(G')$ is at most $k+2$.

Lemma 5. The domination number of G is at most the domination number of G'.

Theorem 3. Given a general graph G and an integer k (as part of the input), deciding whether $\sigma_4(G) \leq k$ is NP-hard.

Moreover, we provide an approximation result via the Set Cover problem.

Definition 6. Set Cover: Given a universe $\mathcal{U}$ and a collection $\mathcal{C}$ of subsets of the universe, decide if there exists a cover, i.e. a subcollection $C' \subseteq \mathcal{C}$ such that $\bigcup_{c \in C'} c = \mathcal{U}$.

We create an instance of Set Cover, where $\mathcal{U}=V(G)$ is the universe and

$\mathcal{C} = \{N[v] : v \in V(G)\}$ is the collection of subsets. Notice that a solution to this instance is exactly a solution to the dominating set problem. We produce an lnn-approximation for this instance by using the greedy algorithm for set cover [6] and then use it in the following theorem as a tool to obtain a guarding strategy.

Theorem 4. There exists a polynomial-time $2lnn$-approximation for $\sigma_2(G)$.

5. Conclusions

5.1. Open Questions

Starting with the fractional robber case, we would like to know whether there exists a relaxed variant of Cops & Robber that can provide an approximation for the cop number. Moving to the fast robber on the grid case, the basic question remains: what is the number of cops needed to capture the fast robber on the $n \times n$ grid? Overall, the big question remaining open is Meyniel's $\sqrt{n}$-conjecture.

For the Surveillance game, one could examine what happens when $\Delta = 4$ or $\Delta = 5$. The cost of connectivity also remains open. Finally, it would be interesting to search for an approximation or an inapproximability result about $sn(G, v_0)$.

Regarding Eternal Security, the complexity is still open. One could delve into combinatorial (graph-theoretic) issues as well.

5.2. In A Nutshell

Many combinatorial games have appeared in mathematical literature over the course of years. One needs to highlight the correlation between combinatorial games and graph theory, since any problem regarding a combinatorial game can be reduced into another problem of graph-theoretic nature. Moreover, the complexity of deciding such games seems to be hard to establish. Last, combinatorial bounds tend to be hard to prove, since the nature of these games makes things complicated even for small graphs.

A broader future research direction would be to try to create new combinatorial

games that model other optimization problems. Perhaps, new insights could be derived by examining them in a game context. Finally, the range of applications could be extended such that certain combinatorial games acquire a more pragmatic nature that counterbalances the abstract theoretical framework in which they are studied.

References

[1] M. Aigner and M. Fromme, A game of cops and robbers, Discrete Applied Mathematics, vol. 8, pp. 1-12, 1984.

[2] M. Anderson, C. Barrientos et al., Maximum demand graphs for eternal security, Journal of Combinatorial Mathematics and Combinatorial Computing, vol. 61, pp. 111-128, 2007.

[3] A. Bonato, P. Pralat, C. Wang, Pursuit-Evasion in Models of Complex Networks, Internet Mathematics, vol. 4, pp. 419-436, 2007.

[4] A.P. Burger, E.J. Cockayne et al., Infinite order domination in graphs, Journal of Combinatorial Mathematics and Combinatorial Computing, vol. 50, pp. 179-194, 2004.

[5] E. Chiniforooshan, A Better Bound for the Cop Number of General Graphs, Journal of Graph Theory, vol. 58, pp. 45-48, 2008.

[6] S. Dasgupta, C.H. Papadimitriou, U.V. Vazirani, Algorithms, McGraw-Hill, 2006.

[7] F. Fomin, F. Giroire et al., To satisfy impatient web surfers is hard. FUN, pp. 166-176, 2012.

[8] P. Frankl, Cops and robbers in graphs with large girth and Cayley graphs, Discrete Applied Mathematics, vol. 17, issue 3, pp. 301-305, 1987.

[9] M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete problems, Proceedings of the sixth annual ACM symposium on Theory of computing, pp. 47-63, 1974.

[10] F. Giroire, D. Mazauric et al., Connected Surveillance Game, SIROCCO, 2013.

[11] F. Giroire, I. Lamprou et al., Connected Surveillance Game, Theoretical Computer Science, 2015.

[12] F. Giroire, N. Nisse, S. Pérennes, R.P. Soares, Fractional Combinatorial Two-Player Games, Manuscript, 2013.

[13] J. Goldwasser and W.F. Klostermeyer, Tight bounds for eternal dominating sets in graphs, Discrete Mathematics, vol. 308, pp. 2589-2593, 2008.

[14] W. Goddard, S.M. Hedetniemi, S.T. Hedetniemi, Eternal security in graphs, Journal of Combinatorial Mathematics and Combinatorial Computing, vol. 52, pp. 169-180, 2005.

[15] A.S. Goldstein and E.M. Reingold, The complexity of pursuit on a graph, Theoretical Computer Science, vol. 143, pp. 93-112, 1995.

[16] M.A. Henning and S.T. Hedetniemi, Defending the Roman Empire: a new strategy, Discrete Mathematics, vol. 266, pp. 239-251, 2003.

[17] W. Kinnersley, Cops and Robbers is EXPTIME-complete, Submitted, 2014.

[18] W.F. Klostermeyer and C.M. Mynhardt, Vertex covers and eternal dominating sets, Discrete Applied Mathematics, vol. 160, pp. 1183-1190, 2012.

[19] A. Kosowski, B. Li, N. Nisse, K. Suchan, k-Chordal Graphs: from Cops and Robber to Compact Routing via Treewidth, ICALP '12, track C, Springer LNCS 7392, pp. 610-622, 2012.

[20] L. Lamport, R. Shostak, M. Pease, The Byzantine Generals Problem, ACM Transactions on Programming Languages and Systems, vol. 4, pp. 382-401, 1982.

[21] L. Lu and X.Peng, On Meyniel's conjecture of the cop number, Journal of Graph Theory, vol. 71, pp. 192-205, 2012.

[22] R. Nowakowski and P. Winkler, Vertex-to-vertex pursuit in a graph, Discrete Mathematics, vol. 43, pp. 235-239, 1983.

[23] A. Quillot, Some results about pursuit games on metric spaces obtained through graph theory techniques, European Journal of Combinatorics, vol. 7, pp. 55-66, 1986.

[24] C. S. ReVelle and K. E. Rosing, Defendens Imperium Romanum: A Classical Problem in Military Strategy. American Mathematical Monthly, issue 107,

pp. 585-594, 2000.

[25] A. Scott and B. Sudakov, A bound for the Cops and Robbers problem, SIAM Journal on Discrete Mathematics, vol. 25, pp. 1438-1442, 2011.

[26] K. Sugihara, I. Suzuki, On a pursuit-evasion problem related to motion coordination of mobile robots, 21st Annual Hawaii International Conference on System Sciences, vol.4, pp. 218-226, 1988

Performance Evaluation of an Enhanced x86 Microprocessor Model with Data and Instruction Cache Prefetchers

Fotini-Maria K. Bratsaki (marifaih@di.uoa.gr)
George Ath. Papadimitriou (georgepap@di.uoa.gr)

Abstract

Modern microprocessor architectures have shown demonstrable progress in performance through the last decades. The increase in the rate of microprocessor clocks extends the computing capabilities but at the same time they significantly exceed the clock frequency of the main memory. Data and instruction access penalties become the main performance bottleneck in high-performance computing and increase processor stall time. Data prefetching is used in order to improve the memory system throughput. This thesis shows how prefetching can effectively change the processor performance by modifying the MARSSx86, a state-of-the-art full system simulator for x86 micropro-cessors. More specifically, we integrate a stride prefetcher for first-level data cache and a sequential prefetcher for first-level instruction cache. The results, using the SPEC CPU2006 benchmark suite, with and with-out hardware prefetching mechanisms, indicate that stride prefetching on a scalar processor can significantly decrease the cache miss rate of particular programs.

Keywords: Micro-Architectectural Simulation, Benchmarks, Optimization, Prefetching

Advisor

Dimitris Gizopoulos, Professor

1. Introduction

Simulation is the imitative operation of a real-world process or system over time. It is used to find a cause of a past occurrence or forecast future effects of assumed circumstances without the cost and possibility of risk that an actual prototype implementation of a system bears [1]. This is the reason why simulation is extensively used in many cases such as simulation of technology for performance optimization, safety engineering, testing, training, education, and video games. Scientific modeling of natural systems or human systems is simulated in order to gain insight into their functionality. Simulation is also used when the real system cannot be engaged, because it may not be accessible, or it may be dangerous or unacceptable to engage, or it is being designed but is not yet built, or it simply does not exist.

In order to comprehend the importance of simulation software, one has to observe its applications. Simulators are widely used in several scientific fields, including computer architecture and many research fields almost exclusively depend on them. The growing urge of accurate tools has led to a lot of study and research around simulation techniques. As a result we have a constant improvement of existing simulation tools and development of many new ones [2]. Through simulators researchers can evaluate different hardware designs without building costly physical hardware system prototypes. They can access non-existing computer components or systems and obtain detailed performance metrics, as a single execution on simulators can often generate a large set of performance data. In addition to this, debugging on real hardware typically require re-booting and re-running the code to reproduce the problems. Thus, this can be avoided as some simulators that have a fully controlled environment, allow the developers to run code backward once an error is detected.

The simulation results are often used for validation, debugging, performance met-rics in research projects and for alternative implementations' evaluation without actually building the systems in real life. This is the prevailing design space exploration method for new architectures and microarchitectures. Accuracy of simulation models provides significant support to the design of correct and efficient computing systems. Actually, the simulator itself is a program that produces results such as input/output signals and statistical data functionality and performance of the given model (execution time, cycles etc.)

Depending on their purpose and level of detail, architectural simulators can be cat-egorized as functional and performance simulators. “Functional simulators” empha-size on achieving the same function as the machine we want to simulate, while “performance simulators” aim to accurately reproduce the performance behavior, in addition to functionality. Therefore, performance simulators usu-ally maintain a more detailed description of the initial machine and in most cases deliver much more detailed results. Performance simulators are divided to cycle accurate simulators and instruction schedulers.

The first category includes the simulators that describe the microarchitecture of a processor on a cycle-by-cycle basis, maintaining data that represent the state of all components at any time during the simulation. Then, the simulators are subcategorized depending on the kind of input that they accept, such as “Trace driven” and “execution driven” simulators. The first category consists of simulators that read an instruction trace with given input that has been already produced by a previous execution. The second one has simulators that read a file with ex-ecution code and runs the program like it would be run on a regular system. The first category tools can be easier developed and they can produce faster simula-tion results. The second category of simulators let us execute any program that has been translated for the given architecture and produce results that are more accurate (closer to reality of an actual implementation) as it takes into considera-tion the system resources that form the complete execution path [5].

Instruction schedulers only simulate an instruction-set architecture and lack the technical details of specific implementations. “Instruction scheduler” simula-tors are usually faster but are not capable to accurately emulate implementa-tion techniques that exist below the architectural level (i.e. micro-architectural mechanisms), such as pipelining and out of order execution. They often serve in simulating older hardware devices, especially in cases where timing information of the modeled design is not required. On the other hand, “cycle-accurate simu-lators” constitute the category of architectural simulators with the best accu-racy. More specifically, they emulate a computer system at the microarchitecture level. By the term “microarchitecture” we refer to the way a given instruction set architecture is implemented on a processor. This level of detail gives the ability to accurately model a processor design exactly as implemented, taking into account the timing information of possible pipelining stages, multiple functional units, inter-cluster communication bandwidths, micro-operations, out-of-order models,

non-blocking memories and other similar details that may exist under the layer of instruction set architecture. Depending on the required accuracy of results, the detail of the subject model may vary.

2. MARSSx86 - A QEMU-Based Full System Simulator

MARSSx86 (abbreviation for Micro-ARchitectural and System Simulator for x86-based Systems) is a framework for fast cycle accurate full system simulation of the x86-64 architecture of single core, multicore and heterogeneous core configurations. It is open source and it is built on top of QEMU emulator in order to support full system simulations running unmodified operating systems and applications. It is a simple yet efficient full system architectural simulation environment based on tools that already exist [6]. It is imperative to have a full system simulation tool that incorporates realistic simulation models for other system level components such as the chipset, DRAM, network interface cards and peripheral devices in addition to accurate simulation models for single and multicore processors implementing the x86 instruction set architectures. Such a tool is useful for evaluating and developing products that will use current and emerging single and multicore x86 chips.

MARSSx86 is built on top of QEMU: a stable emulation framework which consists of various components like a CPU emulator, memory management unit, I/O devices and chipsets. It provides a multicore simulation environment for x86-64 ISA, with detailed pipeline model, including the breakdown of instructions into micro-operations (uops) per second. The simulated processor model used in MARSSx86 is derived from PTLsim, which simulates cycle-accurate out-of-order cores but also has some extensive enhancements for improved simulation accuracy and performance.

One of the most important characteristic of MARSSx86 is the support of flexible switching between the simulation environment and the native x86 emulation mode of QEMU. This is done by permitting the fast-forwarding of simulation in the emulation mode to a specific amount of instructions where cycle-accurate simulation is needed. Similarly, unmodified operating systems can be booted on MARSSx86 and the execution of unmodified x86 binaries of applications and existing libraries can be simulated on it. MARSSx86 includes detailed and cycle-

accurate models of modern memory hierarchy for single-core and multicore architectures. The latter also supports coherence in cache hierarchy. Last but not least, this microprocessor simulation system permits system-level data, such as network packet images and disk block images, to be imported into the simulator from the underlying emulated system.

3. Components Design Implementation

3.1. Data Prefetcher

The designed data cache prefetcher implements a stride prefetching mechanism that is trained by the streaming of physical addresses that come from the address generation units. Stride prefetching is based on the idea that detecting some patterns with a constant stride in the executing program can actually improve future prefetching. A stride memory reference pattern is when a sequence of memory addresses is separated by a constant stride “s” as shown in figure 1. This can occur for example when an array is referenced by a loop index.

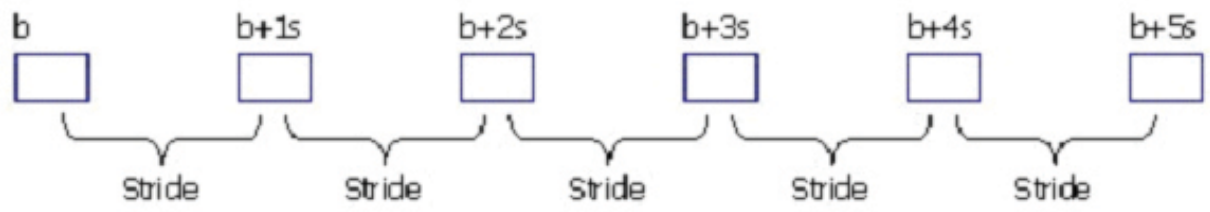


Figure 1: Strided Access

Stride prefetching exploits some loop structures in the executing code by monitor-ing successful load and store instructions. That way capacity misses are eliminated. There is a buffer named miss-stride buffer that is mainly used to eliminate conflict misses. When a memory address miss happens after N other misses, the memory ad-dress is likely to miss again after N other misses. By using the above scenario, more effective predictions can be done. A lot of ways have been proposed to detect a con-stant stride pattern, either more aggressive or less.

When a constant stride is detected, the scheme indicates a successful prediction and triggers new prefetch requests based on this constant stride. This procedure is called prefetcher training. Prefetching then continues as long as the step on the memory address reference for that specific load or store instruction is the same as the previous. Stride prefetching keeps track of the addresses generated for load and store instructions, so that comparison and calculation of the strides for future prefetching can be done. Moreover, the address references that correspond to their instructions must be saved, in order to make the stride detection possible, especially on loop structures with more than one memory references. For saving the address references and their respective instructions, stride prefetching uses a cache named Prefetcher Table or Reference Prediction Table (RPT) where the recent instruction identity information and its address references are stored. Hence, the prefetches are correctly calculated.

The organization of the Prefetcher Table is shown above in figure 2. The table is indexed by the CPU's program counter (RIP). When memory instruction mi is execut-ed for the first time, an entry for it is made in the PT with the state set to initial signi-fying that no prefetching is yet initiated for this instruction. If m_i is executed again before its PT entry has been evicted, a stride value is calculated by subtracting the previous address stored in the RPT from the current effective address.

The state of the PT is incremented during the executions upon sequential policies by correctly handling strided array references. It must be noted that, prefetch addresses are calculated as following:

$$effective\ address + (stride \times distance)$$

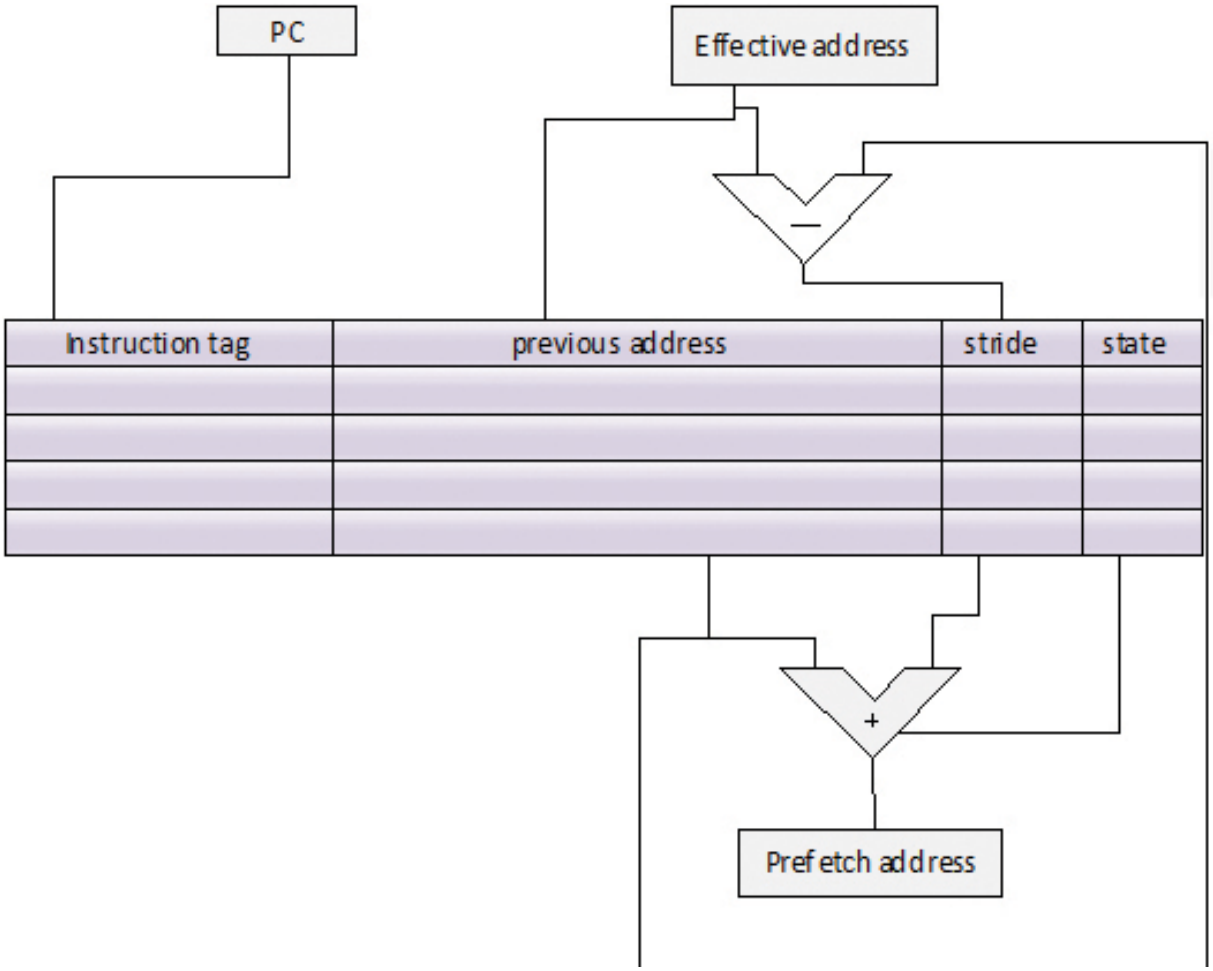


Figure 2: The organization of the prefetcher table

In order to efficiently exploit and incorporate the new distance attribute, it is taken into account the RIP as well as the Program Counter Look Ahead (PC-LA). This pseudo Program Counter initiates prefetches and the difference between the RIP and LA-PC is the distance δ. Prefetch requests are sent to a buffer that keeps a prefetch queue before sending them to a single data cache. A Prefetch Table is used in the data cache prefetcher that stores all the data used for training. The Prefetch Table has a valid bit field that shows if the entry is valid or not, a past physical address used that stores the previous address reference and a register instruction pointer of the given x86 instruction used as a tag. In addition, there are two fields: the stride and the confidence, which store the stride value in order to be compared with the new reference, and the confidence value that shows the aggressiveness of the algorithm respectively. Finally, there is an LRU field for the associatively array implementation. The physical addresses that are generated provide data to the training units of the prefetcher. This procedure is the

prefetcher notification, and helps allocate the entry that is located in the tail of the prefetcher's input buffer. The oldest buffer entry is sent to the Prefetch Table where a set is selected by using the RIP's least significant bits as an index number.

3.2. Instruction Prefetcher

The instruction prefetcher uses a prefetch-on-miss sequential prefetching technique that is trained when an instruction cache miss happens. Sequential prefetching is a method that prefetches data according to their location, which means that relies on spatial locality. Cache fetching is done in blocks, named cache blocks. Multiple word cache blocks are themselves a form of data prefetching and grouping memory words into single units is a method that exploits spatial locality and prefetches data that are going to be referenced in the near future. The degree to which large cache blocks can be effective in prefetching data is l imited by the ensuing cache pollution effects. Cache blocks are fetched by the processor based on the One Block Lookahead (OBL) method. That way, the block B_{n+1} is fetched when B_n is accessed.

OBL implementations differ depending on what type of access to block b initiates the prefetch of b+1. Other approaches depend on what type of access to block n trig-gers the Prefetch of n+1: Prefetch-on-miss, Tagged Prefetch, and Sequential Prefetch with K=2, where K shows the degree of prefetching. One of them, the prefetch-on-miss algorithm simply initiates a prefetch for block b+1 whenever an access for block b results in a cache miss. If b+1 is already cached, there is no memory access.

As in data prefetcher, the input is placed in the input queue and a prefetch request is initiated aiming at the next sequential instruction cache line. Finally, it is deployed in the instruction prefetch queue and the request is sent to the instruction Level 1 cache. In case a miss happens, an entry named Miss Address Buffer is allocated and the request is sent to L2 cache. If an entry is already on the MAB then the requests are dropped.

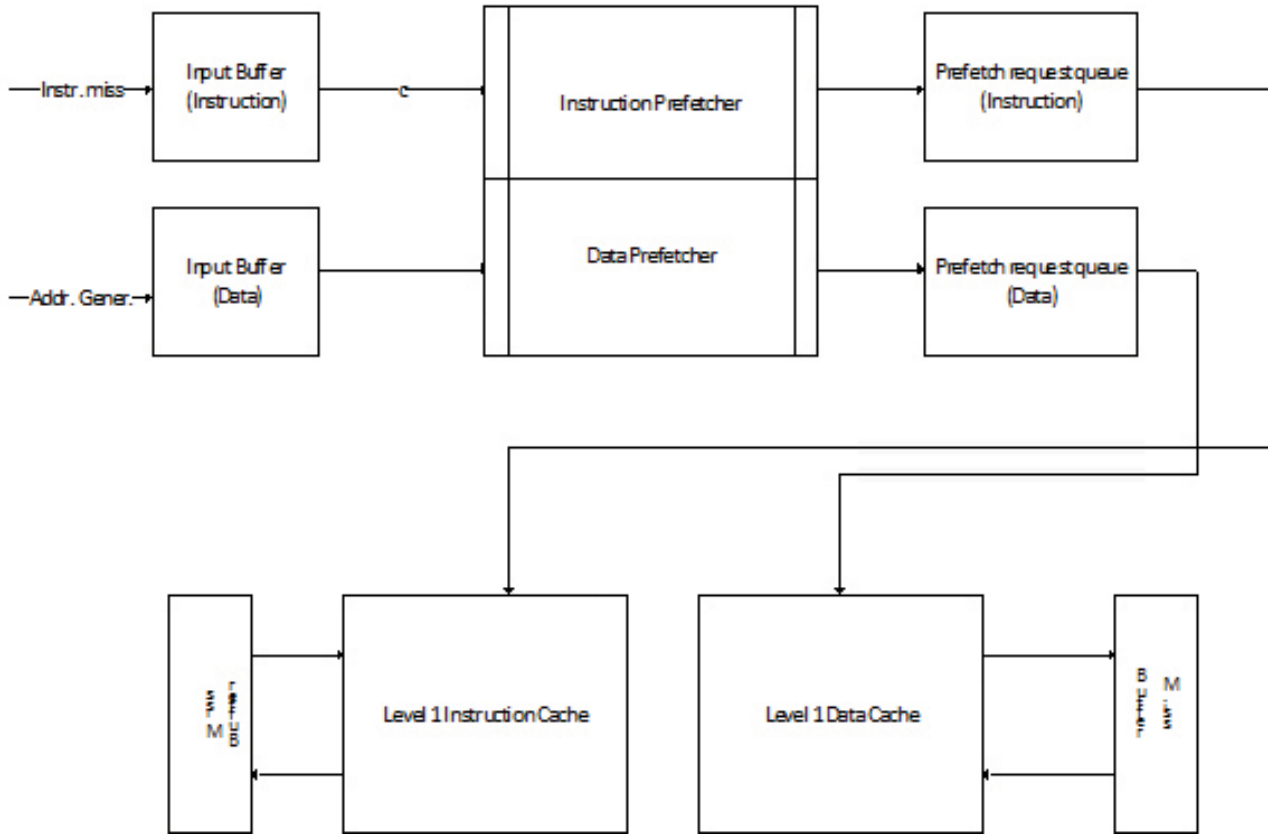


Figure 3: Design with the Prefetching Components

3.3. Design Summary

To begin with, according to the implementation, instruction and data prefetchers are an entity and can share some functional parts but they are internally separated and, in fact, they can be considered as two independent components. For instance, it can be seen that each one of the components, has a separate queue for the requests. More specifically, prefetch requests that are in the queue are treated equally.

The implementation consists of two input buffers, two prefetch (or output) queues, the one for instruction prefetching and the other for data and one Prefetch Table with a configurable size. The size of input and prefetch queue is also configurable. The input buffer entry has a memory address reference, an instruction pointer, a thread id in case of multicore systems. Each prefetch table entry has, as mentioned before, the last memory address, an instruction pointer named RIP (register instruction pointer), the recorder stride and the confidence

field. The instruction pointer is used as an entry tag. It is noteworthy that, when the input queue is full the new requests are dropped, while the prefetch queue drops the first request that came in, as it uses the first in-first out method. As far as the data prefetcher is concerned, the stride prefetching scheme is used. The figure 4 below shows the basic structure of the implementation.

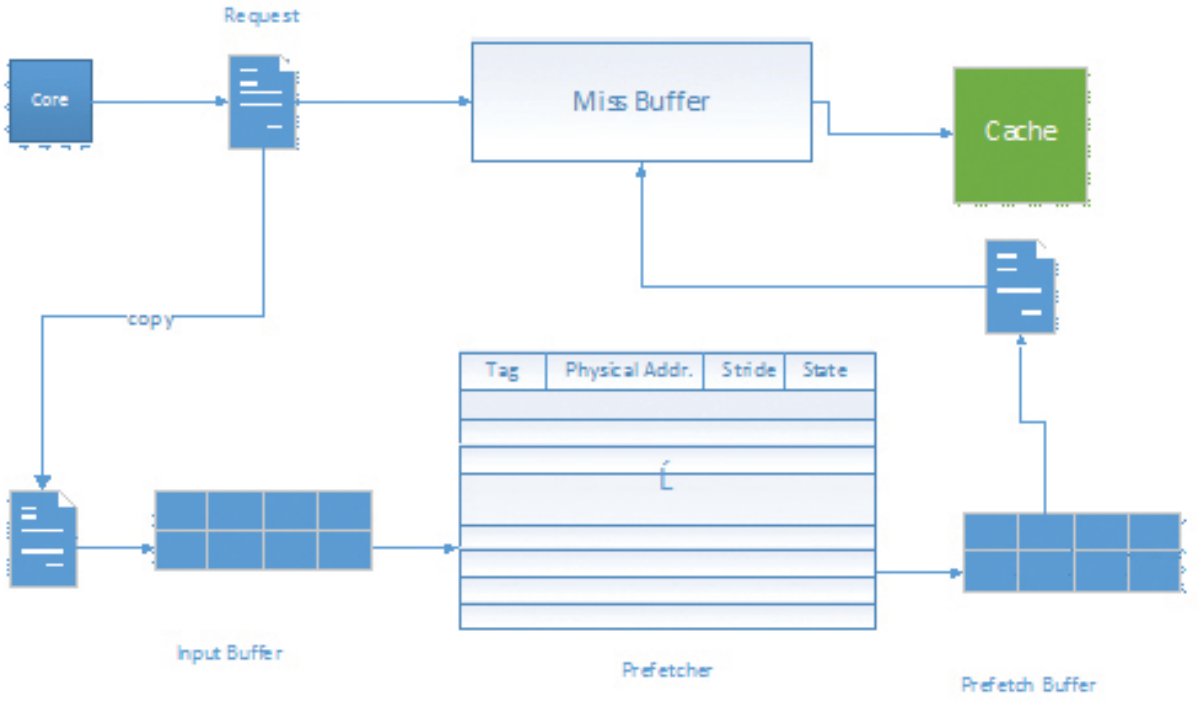


Figure 4: Prefetcher design

When a request gets into the prefetcher, the current RIP field is probed to the prefetch table, in order to check if there is already an entry with the same values. If there is not an entry with the same RIP, then we have a miss so a new entry is inserted to the prefetch table with the request's data in the entry's fields.

If there is an entry with the same RIP, then we have a hit in the prefetcher table. Consequently, the stride is calculated. Stride is the difference of the request' s physical address (new) and the entry's physical address (old). If they are the same, then, we have a NULL stride exception. As a result, the old physical address is overwritten by the new one. If the physical addresses are different their difference must be in the allowed limits that are defined for strides. If the stride is over the limits then we have an exception. If not, the request's stride (new) is compared to the value of the prefetcher table's stride (old stride). If the new stride is different than the old one and the state field (confidence) is

less than the threshold, then the previous stride is updated to the new value of the stride. The entry found on the prefetch table is overwritten with the values of the request.

If the new stride is different than the old one and the state field is greater or equal to the threshold, then the state field is decreased by one. Consequently, the entry found on the prefetch table is overwritten with the values of the request. If the previous stride is equal to the new stride and the state is less than the threshold, then the state is increased by one. If the previous stride is equal to the new stride and the state is greater or equal to the threshold, then there is a pattern found and after the state is increased by one, and an output is produced, then the prefetcher queue entry fields are updated with the request's data. The output has the physical address increased by the stride value multiplied by the algorithm's degree. This means that the prefetch table has predicted an entry according to the data pattern that was detected. Once again, the found entry is overwritten with the values of the new request.

The degree value that defines the level of prefetching is configurable. This degree allows more than one data to be prefetched. For instance, if stride=5, last physical_address=3000 and degree=3, then the next data that will be prefetched will be in the addresses: 3010, 3015, and 3020.

4. Simulation Results

We have performed a series of simulation for performance evaluation with SPEC CPU2006 benchmarks [14]. Twenty-four out of total 29 benchmarks were tested successfully in our experiments. We have excluded the other five benchmarks (bwaves, gobmk, gromacs, dealII and soplex) that had problems and did not finish the test. The tables below present the simulation data that show the performance behavior of the benchmarks for 1 billion total committed instructions. Almost in every case in figure 5, the Experimental model shows an improvement of at most 22% compared to the Baseline model. The amount of instructions that were chosen for simulation may not be a large part of each benchmark. Nevertheless, they are acceptable for turning adequate conclusions about prefetching. It is known that at the beginning of any program there are lots of initialization procedures and that at the ending we usually use store/write

instructions. In reality, prefetches that are made just before the end of the program will not be used as the instructions will probably store, print or return a certain value.

The figure 5 below indicates that each benchmark has different structure and behavior and this means that they produce different results as far as prefetching is concerned. Finally, these findings suggest that prefetching cannot always be productive at the same level for all programs. Correspondingly, the results of the simulations are graphically presented below. The values of Instructions per second (IPC) and speedup for 1 committed instructions are shown below.

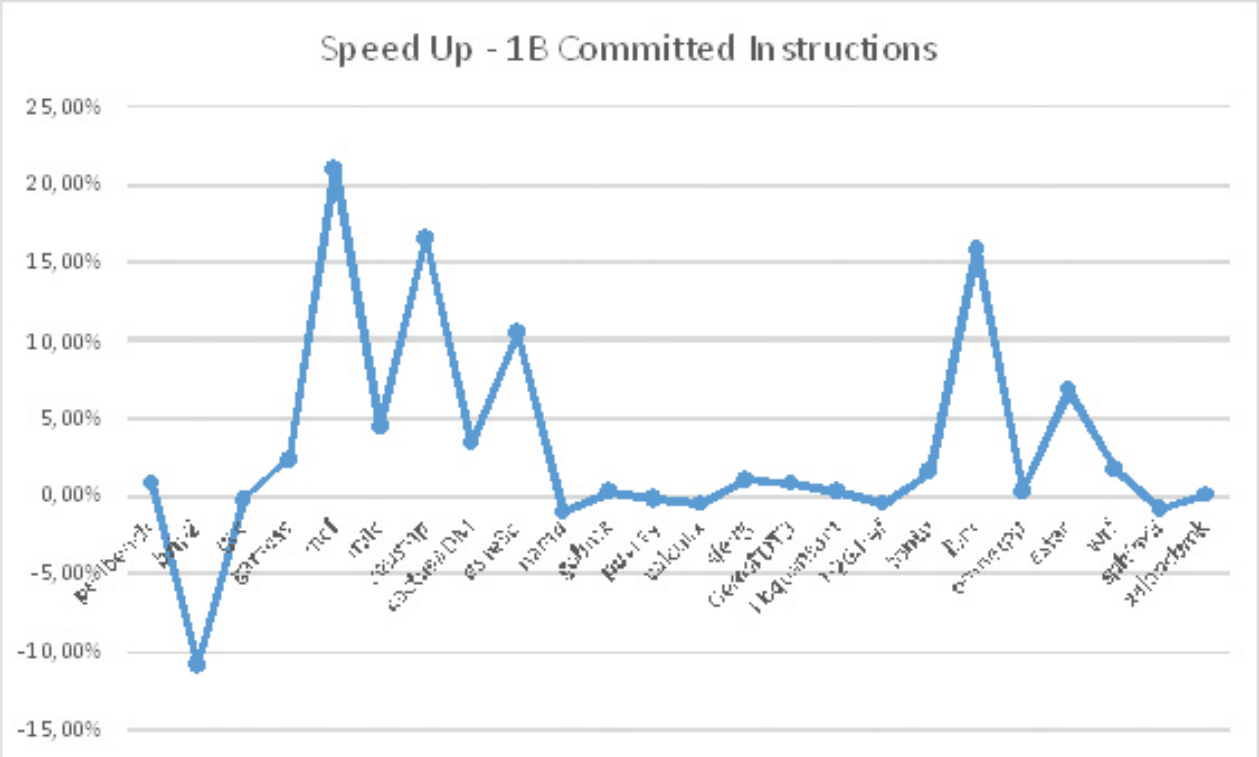


Figure 5: Speed Up - 1B Committed Instructions

5. Conclusion

This work describes the modification of a state-of-the-art x86 full system simulator and more specifically the integration of new a new memory controller in the existing memory hierarchy: a stride prefetcher for data caches. We

successfully managed to satisfy the goal of the new implementation that is to increase speedup during the simulation by using prefetching and by integrating two L1 prefetcher components (data and instruction).

During this work, it was found that stride prefetching may not be possible to establish an access pattern for an instruction that uses an indirect address because the instruction may generate effective addresses which are not separated by a constant stride. In addition to this, the algorithm is far less efficient at the beginning and at the ending of a loo. This means that, prefetches are initiated only when an access pattern is discovered, which means that for at least the first two iterations there will be no prefetching.

Furthermore, the end of prefetching in a loop requires an incorrect prediction in order for the subsequent prefetching to stop. As a matter of fact, the speedup may vary depending on the benchmark's structure like loops and the algorithm's level of aggressiveness. Moreover, it is important to say that some programs may respond better to this prefetching scheme, as they can be highly vectorized.

Given these points, there are some benchmarks that are prefetch-friendly, which means that in some cases as shown in the experimental results section, they can benefit from prefetching. On the other hand, there are some others that their structure and behavior does not benefit from prefetching and show degraded performance.

References

[1] «Computer architecture simulator» http://en.wikipedia.org/wiki/Computer_architecture_simulator. [Access 20/8/2014]

[2] A. Gupta, J. Hennessy, K. Gharachorloo, T. Mowry και W.-D. Weber, Comparative Evaluation of Latency Reducing and Tolerating Techniques, Toronto, Cana-da: ACM New York, 1991.

[3] A. J. Smith, «Cache Memories» τόμ. 14, αρ. Issue No 3, 1982.

[4] A. Patel, F. Afram, S. Chen και K. Ghose, «MARSSx86: A Full System Simulator for x86 CPUs» New York, 2011.

[5] N. Foutirs, D. Gizopoulos, A. Chatzidimitriou, J. Kalamatianos και V. Sridharan, «Performance Assessment of Data Prefetchers in High Error Rate Technologies» 2014.

[6] H. Kang και J. L. Wong, «To Hardware Preeftch or not to Prefetch» 2013.

[7] S. P. VanderWiel και D. J. Lilja, «Data Prefetch Mechanisms» 2000.

[8] A. Hatzidimitriou, PTLsim x86 architectural simulator: extending the data and instruction prefetching subsystem, Athina: Panepistimio Pirea - Tmima Pliroforikis, Septemvrios 2013.

[9] G. Andrianakis, «Epektasi Prosomioti Arxitektonikis Ipologiston SESC» ATHINA, ETHNIKO KAI KAPODISTRIAKO PANEPISTIMIO ATHINON, IANOUARIOS 2010.

[10] D. P. a. L. K. J. Karthik Ganesan, «Generation, Validation and Analysis of SPEC CPU2006 Simulation Points Based on Branch, Memory and TLB Characteristics» 2009.

[11] L. K. J. Arun A. Nair, «Simulation Points for SPEC CPU2006» 2008.

[12] E. P. B. C. Greg Hamerly, «How to Use SimPoint to Pick Simulation Points» 2004.

[13] D. A. Patterson and J. L. Hennessy, «Computer Organization and Design», 2010, ISBN 978-960-461-351-9

[14] Standard Performance Evaluation Corporation, SPEC Benchmarks http://www.spec.org/ [Access 11/8/2014]

Data Mining on Social Media to Detect Events and Disasters

Antonia D. Saravanou (antoniasar@di.uoa.gr)

Abstract

Over the last few years, social media platforms have become more and more popular. Collecting, editing and analyzing big data has become possible with a big range of application programming interfaces (APIs) offered from the most wide-spread social media platforms. Nowadays, many data mining researchers focus on detecting events and emergencies through social media. The goal is to recognize disasters early in order to inform public, allocate resources and special response teams.

In this thesis, we explore the use of Twitter as a mechanism used in disaster relief, and consequently in public safety. In particular, we perform a case study on the floods that occurred in the United Kingdom during January 2014, and how these were reflected on Twitter, according to tweets (i.e., posts) submitted by the users. We present a systematic algorithmic analysis of tweets collected with respect to our use case scenario, supplemented by visual analytic tools. Our objective is to identify meaningful and effective ways to take advantage of the wealth of Twitter data in crisis management, and we report on the findings of our analysis.

Keywords: Text Mining, Web Mining, Event Detection, Emotion Extraction, Clustering Techniques, Visualization, K-Means, V-Analytics

Advisor

Dimitrios Gunopulos, Professor

1. Introduction

Social Media services, such as Twitter, have become a very important source of information for academia, industry and the real content providers, the users. For instance, Twitter now counts more than 271 million monthly active users, with an approximate of 500 million updates posted per day. These updates are called tweets and they are short messages that users are able to post, up to 140 characters. The result is a constant flow of user-generated content, usually referred as the Twitter Stream.

Social Media are complementary to online blogs, where the former contains snippets of more up-to-date information, while online blogs are used for expressing someone's thoughts, ideas, beliefs and are the result of a more thought-through process. The significant advantage of microblogs compared to blogs is their constant updating. This characteristic of fast change, along with the imposition of an upper length limit, is what sets microblogs apart from other social networking services. We address the problem of quick detect of an event and the location that the event occurred, the monitoring progress and evolution of the incident in order to early inform citizens and the corresponding authorities.

Unfortunately, automatically identifying real time events from microblogs is not that easy. One of the challenges is: 1) the large adoption means that we must process in real time voluminous amounts of data. 2) The content (text) is short, very noisy, with a lot of slang and personal style, and diverse in numerous ways, regarding location, languages and themes. Finally, 3) the precise location of a user is generally scarce leading to several techniques for location extraction.

Motivated by shortcomings of existing work, we address the problem of detecting events in a stream of short-form messages, focusing on Twitter. In this thesis, we concentrate on analyzing data extracted from social media, in order to identify emergencies. The analysis of the social media's data includes emotion and location extraction. We focus on detecting tweets that contain information about floods that occurred during the period of January 2014 in England. We look for detecting in detail the areas that were affected by the floods, as well as how the floods were spread in the entire area of England. By monitoring the Twitter stream, we can access the reactions of users to an event of the real world. An event is triggered by a significant change in the emotional state of a potentially large group

of people, and our goal is to capture this sudden change.

In the context of this thesis, we focus on identifying emergencies, such as the floods during January 2014 in England, using data from Twitter, we seek to detect the areas affected by the floods. We propose techniques to correlate the characteristics of messages with the event and evaluate the quality of results. Also, using appropriate visualization tools we can identify the exact area of the incident. The goal is to recognize disasters early in order to inform public, allocate resources and special response teams.

Overall our system integrates the following distinct facets:

- ***Event Detection***, we extract tweets that are related to a given event type from the raw data, crawled from Twitter API during the period that the event occurred.
- ***Emotion analysis***, grounded on influential notions from affective and cognitive theories from psychology. Sentiment analysis, text analytics and classic data mining methods are used for this purpose.
- ***Location Extraction***, placing tweets and the visualization of a clustering technique on a map.
- ***Area Separation***, using a clustering technique, separate the entire crawled area on smaller ones, in order to find how the event affected smaller geographical areas.

2. Background

The nature of social media data has led to an increased use of the medium for the purposes of event identification. Early works focus on events of specific types e.g. news [9] or political debates and elections. This is typically done by either whitelisting users (e.g., news reporting agencies) or by building on the premise that the topic is inherently polarized, in which case we can monitor the reactions of user communities. Emergency events, such as the ones we consider here (i.e., floods, fires), are not polarized, rendering such techniques useless. On the other hand, whitelisting will only provide access to the raw data – which we can retrieve through other means as well –, without offering any additional insights.

The most characteristic (and successful) example of event identification through social media information is [8]. This work focuses on earthquakes and its main objective is to accurately extract the location of an earthquake. The authors rely on a manually crafted lexicon to match geotagged tweets related to the event. They propose a model that incorporates established scientific theories on wave propagation, through which they identify the origin of an earthquake. These particular theories, however, do not apply to floods, rendering these techniques inapplicable for our use case.

Type-independent event identification has also been a hot research area recently, to avoid the manual or semi-automatic generation of lexicons [1, 2, 5]. These techniques are more complex than the one we propose in this paper, but their objective is in identification and much less in understanding the spread of information on social media or how an event is portrayed online. Although event identification is clearly the first step to disaster management and relief, we believe that our current analysis may provide additional insights on how to utilize social media information.

Recent research attempts have also put social media to the test for flooding events [7, 3]. Such attempts are usually constrained to a statistical analysis of the collected dataset, e.g., #tweets, #users, hashtag distribution, and secondarily to an analysis related to the flooding event itself, e.g., vicinity of tweets to the event. Even in these cases, the evaluation implies that we have access to high quality sensing devices. Although this does not invalidate the findings of the research works, in practice, it would render social media useless for the purposes of flood disaster management.

The most closely related work regarding flooding events is the one in [3]. The authors use their framework to visually examine spatio-temporal data regarding the floods that took place in Germany in the summer 2013. They focus on streamlining a visual analytics workflow that will assist in detecting significant events, and discuss their hypothesis and findings. On the contrary, in our current work, visualization tools are supplementary, as we focus on a methodology and accompanying techniques that will help us identify heavily stricken areas.

3. System Overview

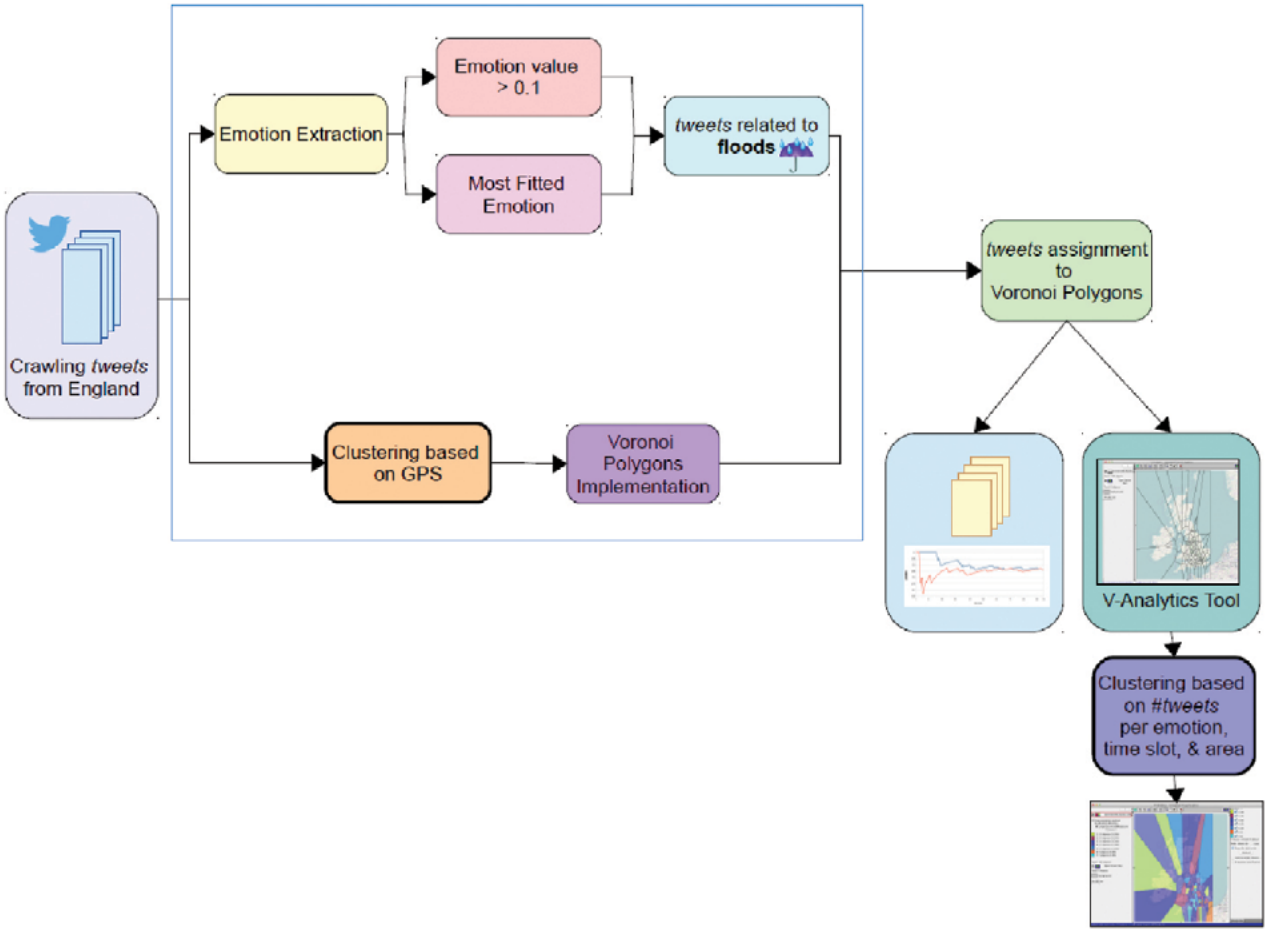


Figure 0: System Overview

In Figure 0, we present an overview of how our event detection analysis and visualization system will work. There are four main phases; the emotion extraction, the search for tweets related to floods, the clustering in order to create smaller geographical areas and the phase that tweets are assigned to geographical areas. In emotion extraction, we use the text of each tweet in order to classify it in one of the emotion classes used. Then, we can choose to set for each tweet the emotion class that has the highest ratio, or the emotion class that has ratio greater than 0.1. Except from finding the most suitable emotion, we need to separate from the raw data, a dataset that contains tweets that have information/comments/thoughts about the flooding phenomenon. In parallel to this, we apply some clustering techniques using the GPS location of all tweets crawled to discover some smaller parts of the whole bounding box used for crawling,

the geographical areas. Afterwards, we build Voronoi polygons with the results from the clustering technique. In addition, we use the datasets generated in the previous phases, depending on the visualization we would like to view. We combine the geographical information with the results of the emotion and "flood context" extraction and create files in compatible type with the V-Analytics tool. Finally, we could use the results either for analysis (timeseries), either for an extra clustering layer in order to discover the progress of the incident, and how it affected each area per time frame.

4. Crawling Data From Twitter (England Area)

We will start by describing the data we collected to perform our case study. Posts on Twitter, generally known as tweets, are short snippets of text, up to 140 characters. Aside from text, they may contain features like (shortened) hyperlinks to external sources, hashtags, i.e., author-generated tags describe the topic of a tweet (e.g. #sports), mentions of other Twitter users, or other data formats, including images and videos. We used our custom built crawler [4] to collect tweets from the Twitter service, using the Streaming API. The Streaming API returns a sample of all public tweets, as these are posted in the service, granting us access to up-to-date information. Given our interest in the floods that took place in the United Kingdom (UK), we limited ourselves to tweets posted from this particular area. To achieve this, we used a bounding box that covers the entire UK, namely [(−13.4139, 49.1621), (1.7690, 60.8547)], and used that as a first filter for collecting the data. This way, the sampling policy is applied directly to those posts that, according to the service, have originated from a location within the requested bounds. In some cases, we may receive a tweet that does not have an associated GPS coordinate, because the service has concluded that it falls within the requested bounding box through other means, e.g. the user's profile. We do not consider such tweets during our analysis, because our approach is based on the specific geolocation of a tweet. More information about the bounding box filter is available at *https://dev.twitter.com/streaming/ overview/request-parameters*.

We applied the data harvesting technique described above for a 5 day period, during early 2014. Once a tweet is received, we store all of its information externally for offline processing. Table 1 summarizes some basic statistics of the

collected data. The entire dataset consists of more than 2.3 million geotagged tweets. The first and last days contain about half of the tweets compared to the rest, as our crawl only covered half a day on each of these occasions.

Table 1: Dataset Statistics

Period	Total #Tweets	#Flood Related Tweets
Jan. 13	351140	2728
Jan. 14	577151	4973
Jan. 15	569108	4159
Jan. 16	578553	4994
Jan. 17	275358	3490
Total	2351310	20344

5. Extraction of tweets related to floods

A problem with our original dataset is that it contains tweets that have been filtered only on the grounds of location. However, because of the high diversity of Twitter users and their interests, it is likely that many tweets will be completely unrelated to the event which we are monitoring. For this reason, we need to perform an additional filtering step, to keep only those posts that are related to floods. To achieve this, we built a custom lexicon, containing tokens related to our event. Note that this is the norm for monitoring events of a particular type in Twitter [8, 9, 10]. We started with a very small seed set of 13 related tokens, including "rain, flood, weather, storm, showers", etc. We parsed the entire dataset and extracted all of the tokens – excluding mentions, i.e., @username – that contain at least one of the seed set keywords as a substring.

Following the methodology described above, we obtained a lexicon with 1546 distinct keywords, much larger in size than the seed set. This lexicon contains the seed keywords in various forms, e.g. raining, floods, #ukweather, etc. However, our approach also yielded a lot of false positives, e.g., brain, train, etc. We carefully reviewed each keyword and discarded everything that is not related to our

use case. The final, cleaned version of our lexicon contains 456 distinct keywords. Although this is a laborious process, it only needs to be performed once.

Table 2: Lexicon Keywords

	Original Lexicon (1546 keywords)		Flood Lexicon (456 keywords)	
Rank	**Keyword**	**In #Tweets**	**Keyword**	**In #Tweets**
1	*rain*	11235	*rain*	11235
2	*train*	6499	*weather*	3331
3	*training*	4593	*snow*	1006
4	*weather*	3331	*raining*	997
5	*brain*	1747	*rainbow*	419
6	*trains*	1251	*storm*	333
7	*snow*	1006	*showers*	273
8	*raining*	997	*rainy*	249
9	*trainers*	813	*flooding*	215
10	*drained*	435	*flooded*	214

Table 2 contains the top-10 keywords for the two lexicons: the one that was compiled directly from the collected tweets, and the cleaned version, containing only the keywords related to our event. It is important to note that keywords with a high number of occurrences (as seen from Table 2) are generic enough and are related to the event type, not to the particular location. This means that we can use these keywords for the same purpose (flood monitoring) in other locations as well.

Algorithm 1 `Selection of Flood-related Tweets`

Input: Set of tweets $\mathcal{T}$, Lexicon $\mathcal{L}$
Output: Set of flood-related tweets $\mathcal{T}_{flood}$

$\mathcal{T}_{flood} \leftarrow \emptyset$;
for every tweet $t \in \mathcal{T}$ **do**
    tokens $\leftarrow$ split $t.text$ into tokens;
    for every token $tkn \in tokens$ **do**
        if $\mathcal{L}$.contains(tkn) **then**
            Add t to $\mathcal{T}_{flood}$;
            break;
return $\mathcal{T}_{flood}$

Having our flood-related lexicon in place, we iterate over the entire collection for a second time. During this step, we select tweets as follows: We tokenize the text of the tweets and keep only those that contain an exact match with at least one of the keywords of our flood lexicon. Algorithm 1 presents the pseudocode for selecting the flood related tweets. Given that, at this point, our lexicon contains only flood-related tokens, we expect the majority of the extracted tweets to be discussing the event of our use case, making them safe to use in our subsequent analysis. We say "the majority" because some lexicon keywords may have a slightly altered meaning on occasions. For instance, the expression "be under the weather" is most likely unrelated to a flooding event, although it contains the seed keyword "weather". To that end, the authors did a preliminary manual analysis of 1000 randomly selected tweets, that contain at least one keyword from the flood lexicon. This analysis showed that 88.5% of the sample is relevant to the event that we care about.

6. Emotion Annotation

We also applied the algorithm described in [1] in order to discover what was the feeling of the user the moment he posted the tweet. This method, uses a classifier to annotate each tweet with rates to 6 different classes of emotions, proposed by Paul Ekman [6]: anger, fear, disgust, happiness, sadness, surprise, plus the neutral state, to describe the absence of an emotion. To do the classification, this method uses dictionaries (affective WordNet) and features of the tweets. As a result, each tweet is annotated with the ratio for each of the 7 emotion classes. In order to annotate each tweet with one emotion, we can annotate each tweet to the emotion that has the biggest ratio, "the most fitted emotion» or we can annotate each tweet to each emotion that has ratio above 0.1. We applied both techniques in the initial dataset and the «flood-related» dataset and conclude in using the «ratio > 0.1» approach for our experiments.

7. Generating Monitored Regions

As we have already pointed out, we are interested in techniques that will provide meaningful insights to our collected data. In particular, we have noted three spe-

cific questions that we would like to answer, through our subsequent experimental analysis:

Q1: How can we identify the areas that have been hit the most by an event?

Q2: How effective can we be in identifying these areas?

Q3: Can we identify areas that have been stroke by the event in a similar manner?

The major concern of question Q1 is the identification of the areas that have been hit by the event the most. We need to identify areas or regions, we need to go beyond the GPS coordinates of a single tweet. We apply k-Means clustering to aggregate the geotagged information of our tweets to form these larger areas, which will create k non-overlapping regions. We extract the geotagging information from all of the tweets of the original dataset. This gives us a little more than 2.3M GPS coordinates in (longitude, latitude) form, covering the entire country (UK). We use the original dataset because it contains a lot more tweets, thus we expect the clusters to better reflect the actual underlying population. Before the clustering step, we convert the GPS coordinates to Cartesian ones, using Mercatorian map projections, for k-Means to work properly with the L2 Euclidean distance. We experimented with different k values (k = 10, 100, 500, 1000) and report on the results for each case.

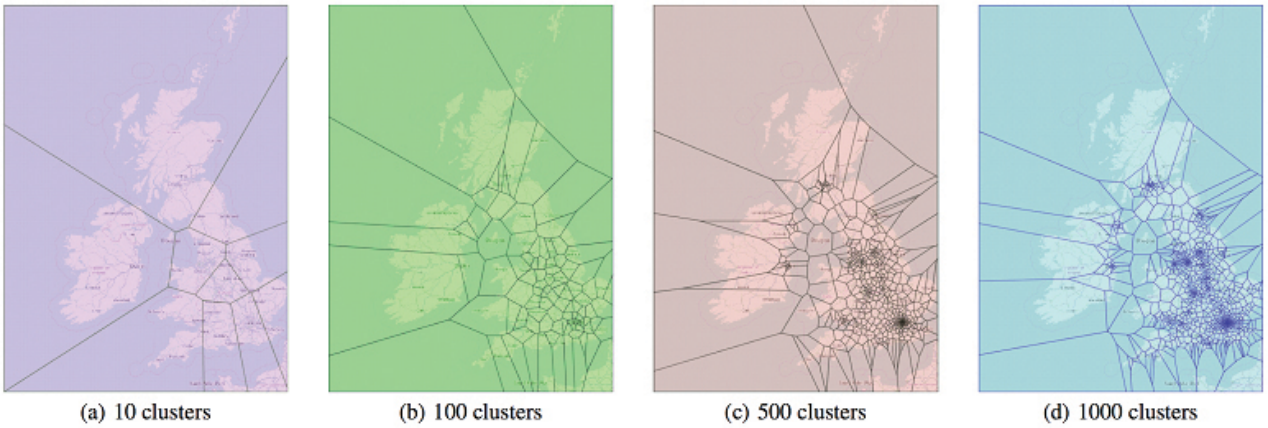


Figure 1: Spatial clusters generated by k-Means clustering on the entire (2.3M) collection of tweets

Figure 1 a)-d) shows the boundaries of each generated cluster as Voronoi polygons, for the different values of k, overlayed on top of a UK map. We can

clearly see that as we increase the number of clusters k, we get more splits in the densely populated areas (London, Manchester, Leeds, Glasgow, etc), as opposed to more rural areas. As a result, our current analysis could be seen as a weak proxy for hierarchical clustering, cutting the hierarchical dendrogram at different levels (heights).

8. Identifying Flood-Affected Areas

To identify the areas that were mostly affected by the floods, we will prioritize the generated regions by their potential of being affected by the flood, and propose 3 schemes to do this. The scheme would practically dictate the order with which emergency response units should attend to each region.

Given that the number of regions to check is quite high (1000 regions at most), the prioritization scheme will also allow us to reduce the evaluation cost. A good prioritization scheme should return highly affected areas first, and less affected areas afterwards. We can then evaluate the top-n regions, as returned by each scheme.

The prioritization approaches that can be used for this purpose are:

- By **#tweets**: This is the simplest scheme and serves as our baseline. The regions are ordered by the number of tweets posted from it, regardless of being flood-related or not.
- By **#flood-related tweets**: The regions are ordered by the number of flood-related tweets posted from that area.
- By **SNR**: A potential problem of the previous scheme is that densely populated areas are more likely to have a high number of flood related tweets. We counter this via a Signal-to- Noise Ratio approach. Each region r is assigned a score

$$score(r) = \frac{\#\,flood\text{-}related\ tweets\ in\ r}{\#\,tweets\ in\ r}$$

and areas are ordered by their respective score.

9. Experiments

9.1. Evaluation

To evaluate our results, we use two independent sources as ground truth information, the monthly hydrological report for January 2014, published by the Centre for Ecology & Hydrology, of the Natural Environment Research Council in the UK (http://www.ceh.ac.uk/data/nrfa/nhmp/hs/pdf/HS_201401.pdf) and the Met Office, published by UK's National Weather Service (*http://www.metoffice.gov.uk/media/pdf/n/i/Recent_Storms_Briefing_Final_07023.pdf*).

The evaluation took place as follows. From the k clusters obtained through k-Means clustering, we select the top-n (n=100 in our case), ordered by each scheme (#tweets, #flood tweets, SNR). We manually review each of the n areas, and compare it against the ground truth information. We use a Likert scale ([1-5]) to specify the degree up to which an area has been affected by flooding (1="not at all", 5="completely flooded"), and assign a score to each of the top-n areas. The assigned score for each area is normalized in the [0-1] range. We then compute and report the running average of the scores up to the i-th ranked area, using the formula

$$value_i = \frac{\sum_{j=1}^{i} \frac{likert\_score(j)}{5}}{i}$$

where $likert\_score(j)$ is the likert evaluation for the j-th region. The result of this evaluation is given in Figure 2. On the x-axis we plot the top-n areas, whereas on the y-axis we plot the running average of percentages, using the above formula. We compare the 3 schemes: i) by number of tweets in the area (All) and ii) by number of flood-related tweets (Flood) and iii) by Signal-to-Noise Ratio (SNR).

We observe that for k=100 clusters, the schemes Flood and SNR behave almost the same, with SNR being slightly better at first. Both techniques perform better than the baseline approach (All), but their differences become blurred after the first 50 areas. Looking at Figure 1(b), we see that there is not much detail in the generated regions, even in areas like London, which might explain the minimal differences among the techniques.

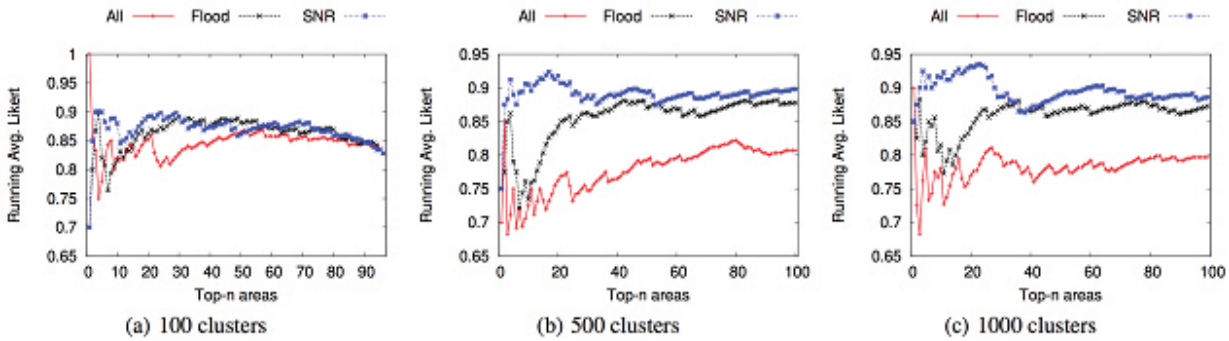


Figure 2: Running average of the normalized Likert scores, of the top-n regions that were selected by each prioritization scheme

When we increase the number of clusters to k=500 or k=1000, Flood and SNR clearly outperform the baseline. This shows that the number of social media users in an area is not a good proxy for the impact of an event. Counting the number of event-related posts (Flood) is a much better approach to identify areas that were hit harder. Nevertheless, Figures 2b)-c) lead us to believe that this technique shares some of the deficiencies of the baseline approach, due to the steep decline early on (around top-15 areas). Unlike the baseline, however, Flood improves much faster. On the other hand, SNR outperforms both of the other techniques, especially during the top-n areas. The scheme's performance, compared to that of Flood, is evidence of our argument that densely populated regions are more likely to have more event-related posts. Therefore, when we account for the number of users in that area, as SNR does, we manage to achieve even better performance. More specifically, SNR is able to maintain a running average of ~0.9 for the top-100 areas, never dropping below 0.85, providing us with quantitative data for question **Q2**. Another thing to note here is that Flood exhibits the same decline during the first top-n regions, regardless of the number of clusters k. This leads us to conclude that this is a result of our collected data, rather than the prioritization scheme.

9.2. Temporal Similarities

One of our final objectives is to identify regions with similar behavior, in the way that the flooding event was perceived by Twitter users. The general implication of such a similarity is that there is an underlying connection, between

these areas. This connection could be at the population level, e.g. the users have similar posting patterns when it comes to such events, or could be due to some other variable, e.g., a nearby river, or a problem in the plumbing system of those areas.

For this type of analysis, we start with our set of k clusters, obtained from k-Means. Each cluster is now a tuple and the output of this analysis will be groups of clusters. Because we are interested in temporal similarities of the regions, the features that we will use will be based on daily information.

We will consider the following sets of features, all of which are now applied to the flood related tweets.

- The number of tweets that were posted each day d from region r, denoted by $count^d_r$.
- The ratio of day d from region r is the fraction of tweets posted that day from that region, over all tweets posted from that region for the monitored period. Formally,

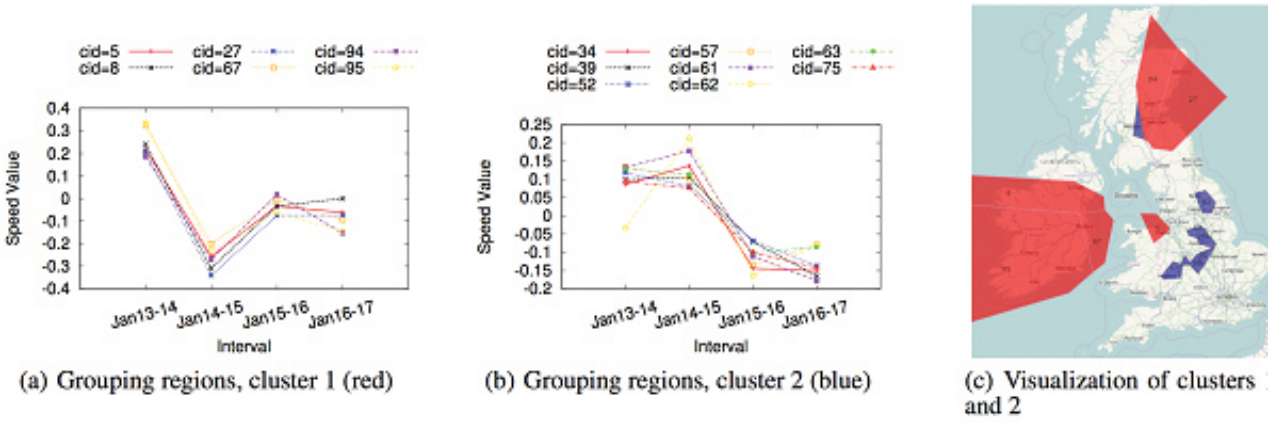
$$ratio_r^d = \frac{count_r^d}{\sum_{\forall d'} count_r^{d'}}$$

- Speed of day d is the difference between the ratio of day d and the preceding one.

$$speed_r^d = ratio_r^d - ratio_r^{d-1}$$

  The rationale of this feature is to capture abrupt changes, and identify areas that were affected without sufficient notice.

We experimented with various combinations of these features, when performing the second level of clustering. Figures 3a-b) show two such groupings that were the result of clustering the regions using only the speed feature. The x-axis refers to the day i for which we evaluate $speed^i_r$, depicted on the y-axis. Each figure refers to a distinct 2nd level clustering, where we can clearly see the difference in the speeds of the regions. Figure 3c) visualizes the two clusterings on the map. Cluster 1 (in red) contains areas from Scotland, Liverpool and Ireland, whereas Cluster 2 (in blue) contains areas mostly from the Midlands.



(a) Grouping regions, cluster 1 (red) (b) Grouping regions, cluster 2 (blue) (c) Visualization of clusters and 2

Figure 3: Second level of clustering information, using the speed feature

Figures 3a-b) illustrate the difference between areas from the two clusters. Areas in cluster 1 are mostly unaffected by floods, whereas cluster 2 contains regions with the opposite behavior. A difference in the trend of precipitation has also been verified with historical data from *http://www.weatheronline.co.uk/*. This is a characteristic example why the speed feature results in one of the best clusterings that we observed.

10. Conclusion

In this paper, we experiment with Twitter, one of the most prominent social media nowadays, for the purposes of disaster management and relief. Using the floods that occurred during January 2014 in the United Kingdom as our use case, we collect geotagged tweets, to analyze them and gain significant insights. We propose a methodology to clean the original dataset, build larger regions to monitor, as opposed to single GPS locations, and process the dataset to identify flood-stricken areas with high accuracy. Our approach, may also be seen as a spatio-temporal tool for discovering the progress of an event on a map and know how these are perceived by the users. In other words, we combined notions from emotional theories with spatiotemporal information, and tackled the problem using a dictionary for the known event. We also showcased our system and We evaluate our findings against ground truth data, obtained from external, independent sources and report on those results. We also suggest features that are better at identifying regions that were affected in a similar way, despite

not being spatially close.

Our current analysis is restricted to a static analysis of the tweets. In future work, we will also focus on the problem of automatically identifying unknown events from the Live Web, as they occur. We would like to develop online or streaming approaches, including the development of online clustering algorithms for our setting.

References

[1] G. Valkanas and D. Gunopulos. How the live web feels about events. In CIKM, 2013.

[2] H. Becker, M. Naaman, and L. Gravano. Learning similarity metrics for event identification in social media. In WSDM, 2010.

[3] M. A. Brovelli, G. Zamboni, C. A. Muñoz, and A. Bonetti. Exploring twitter georeferenced data related to flood events: an initial approach. 2014.

[4] G. Valkanas, A. Saravanou, and D. Gunopulos. A faceted crawler for the twitter service. In Web Information Systems Engineering, WISE 2014, 15th International Conference.

[5] J. Weng and B.-S. Lee. Event detection in twitter. In ICWSM, 2011.

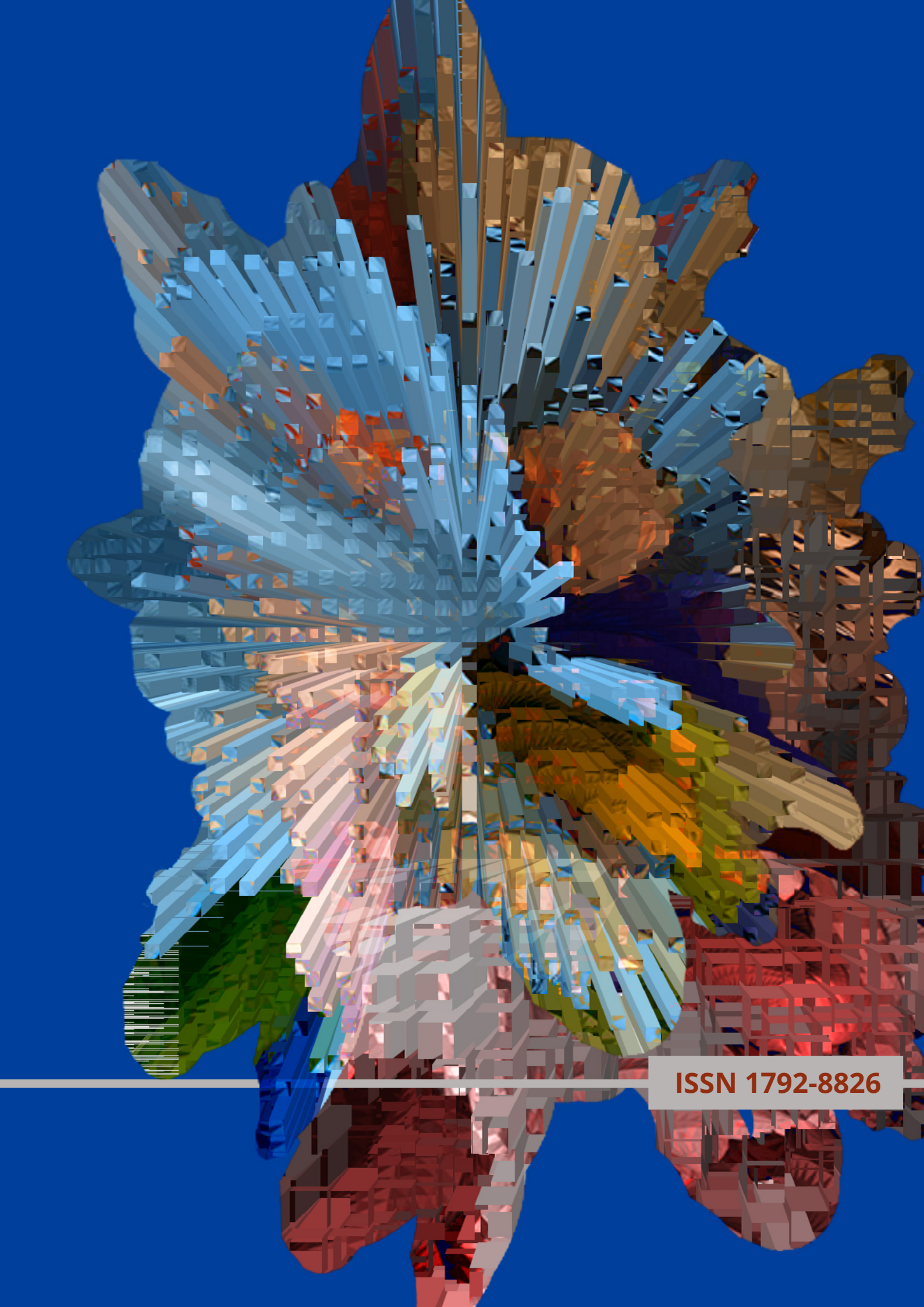
[6] P. Ekman, W. Friesen, and P. Ellsworth, Emotion in the human face: guidelines for research and an integration of findings. Pergamon Press, 1972.

[7] B. Herfort, J. P. de Albuquerque, S. Schelhorn, and A. Zipf. Exploring the geographical relations between social media and flood phenomena to improve situational awareness - A study about the river elbe flood in june 2013. In AGIEL, 2014.

[8] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In WWW, 2010.

[9] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: news in tweets. In SIGSPATIAL-GIS, 2009.

[10] M. Mathioudakis and N. Koudas. Twittermonitor: trend detection over the twitter stream. In SIGMOD, 2010.



ISSN 1792-8826