Πτυχιακή Εργασία

του Νικολάου Καψάλη

Μελέτη υλοποίησης αλγορίθμων με το περιβάλλον MapReduce/Hadoop.

Επιβλέποντες καθηγητές : Μποζάνης Παναγιώτης

Κατσαρός Δημήτριος

Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών Τηλεπικοινωνιών και Δικτύων

Πανεπιστήμιο Θεσσαλίας ,2010-2011

Ολοκληρώνοντας τις προπτυχιακές μου σπουδές στο τμήμα μηχανικών ηλεκτρονικών υπολογιστών, θα ήθελα να ευχαριστήσω όλους εκείνους που συνέβαλαν στην ολοκλήρωση των σπουδών. Πρώτα από όλα θα ήθελα να ευχαριστήσω τους επιβλέποντες καθηγητές μου κ.κ. Μποζάνη Παναγιώτη(Αναπληρωτής Καθηγητής Τ.Μ.Η.Υ.Τ.Δ) και Κατσαρό Δημήτριο(Λέκτορας Τ.Μ.Η.Υ.Τ.Δ.) που μου έδωσαν την ώθηση και τις γνώσεις για να ολοκληρώσω την διπλωματική μου εργασία. Στην συνέχεια θα ήθελα να ευχαριστήσω όλους τους καθηγητές μου που στάθηκαν αρωγοί στις προσπάθειές μου και μου μεταλαμπάδευσαν τις πολύτιμες γνώσεις τους.

Έπειτα θα ήθελα να ευχαριστήσω την οικογένειά μου για την αμέριστη συμπαράσταση(οικονομική και ηθική) όλα τα χρόνια των σπουδών μου. Θέλω να ευχαριστήσω ιδιαιτέρως τον αδερφό μου(Δημήτρη) για την υπομονή που έδειξε όλα τα χρόνια της συγκατοίκησής μας στον Βόλο καθώς επίσης για την αμέριστη βοήθεια και τις γνώσεις που μου μετέφερε καθόλη την διάρκεια των σπουδών μου.

Τέλος θα ήθελα να ευχαριστήσω όλους τους φίλους μου με τους οποίους συνεργάστηκα όλα αυτά τα χρόνια, όχι μόνο τους συμφοιτητές μου, αλλά και ανθρώπους έξω από την σχολή. Δεν θα αναφερθώ ονομαστικά, με τον φόβο της παράλειψης κάποιου.

Στην οικογένειά μου,

και την Φωτίνα.

Περιεχόμενα

ΜΕΡΟΣ Α

1. Εισαγωγή

Το MapReduce είναι ένα προγραμματιστικό μοντέλο και μια εφαρμογή για την επεξεργασία και την παραγωγή μεγάλου όγκου δεδομένων. Τα τελευταία χρόνια παρατηρείται η ανάγκη διαχείρισης όλο και μεγαλύτερου όγκου δεδομένων πληροφοριών(άνθρωποι ανεβάζουν βίντεο στο διαδίκτυο, βγάζουν φωτογραφίες μέσω των κινητών τους τηλεφώνων, αναβαθμίζουν τη σελίδα τους στο Facebook, αφήνουν σχόλια στο διαδίκτυο και πολλά ακόμη) γεγονός που ανάγκασε μεγάλες εταιρίες του χώρου όπως οι Google, Yahoo!, Amazon και Microsoft να αναζητήσουν νέα εργαλεία για την επεξεργασία τέτοιων μεγάλων συνόλων δεδομένων. Η Google πρώτη δημοσίευσε το Map Reduce ένα παράλληλο προγραμματιστικό μοντέλο για την κλιμάκωση των δεδομένων και αμέσως προσέλκυσε το ενδιαφέρον αρκετών εταιριών που αντιμετώπιζαν παρόμοια προβλήματα. Ο Doug Cutting ανέπτυξε μια ανοιχτού κώδικα έκδοση του Map Reduce και την ονόμασε Hadoop. Αμέσως μετά η Yahoo και οι υπόλοιποι κινητοποιήθηκαν και υποστήριξαν αυτή την προσπάθεια.

Σήμερα, το Hadoop είναι ένα σημαντικό κομμάτι της υπολογιστικής υποδομής για πολλές web εταιρίες, όπως η Yahoo, Facebook, LinkedIn και Twitter. Πολύ περισσότερες παραδοσιακές εταιρίες media και τηλεπικοινωνιών αρχίζουν να καταφεύγουν σε αυτό το σύστημα. Αρκετές εταιρίες όπως οι New York Times, China Mobile και IBM χρησιμοποιούν το Hadoop.

Αυτό που έχει σημασία είναι ότι ένας προγραμματιστής σήμερα, πρέπει να γνωρίζει σχεσιακές βάσεις δεδομένων, δικτύωση και ασφάλεια, πράγματα που θεωρούνταν προαιρετικές δεξιότητες δυο δεκαετίες νωρίτερα. Παρόμοια, καλή κατανόηση της κατανεμημένης επεξεργασίας δεδομένων θα αποτελέσει σημαντικό κομμάτι της εργαλειοθήκης του προγραμματιστή στο μέλλον. Πρωτοπόρα Πανεπιστήμια όπως το Stanford και το CMU, έχουν ήδη εντάξει το Hadoop στο πρόγραμμα μαθημάτων της επιστήμης υπολογιστών.

Βιβλιογραφία

Tom White, Hadoop: The Difinitive Guide, California: O'Reilly, 2009

Jason Venner, Pro Hadoop, New York: Springer – Verlag, 2009

Chuck Lam, Hadoop In Action, Connecticut: Manning, 2011

2. Ιστορία του Hadoop

Το Hadoop ξεκίνησε ως ένα κομμάτι του project του Nutch το οποίο με τη σειρά του ήταν ένα υπό project του Apache Lucene. Ο Doug Cutting δημιούργησε όλα αυτά τα τρία projects και καθένα από αυτά ήταν μια λογική ακολουθία των προηγούμενων.

Το Lucene είναι μια βιβλιοθήκη για ευρετηριοποίηση πλήρους κειμένου και για αναζήτηση. Αν μας δοθεί μια συλλογή κειμένων ο προγραμματιστής μπορεί πολύ εύκολα να προσθέσει τη δυνατότητα αναζήτησης στο έγγραφο χρησιμοποιώντας τη μηχανή Lucene. Desktop search, enterprise search και πολλές domain-specific μηχανές αναζήτησης έχουν δημιουργηθεί χρησιμοποιώντας το Lucene. Το Nutch είναι η πιο φιλόδοξη επέκταση του Lucene. Αυτό που κάνει είναι να προσπαθεί να χτίσει μια ολοκληρωμένη μηχανή αναζήτησης στο διαδίκτυο χρησιμοποιώντας το Lucene ως βασικό του συστατικό. Το Nutch περιέχει συντακτικούς αναλυτές για HTML, ένα web crawler, μια βάση δεδομένων συνδεδεμένου γράφου και διάφορα επιπλέον συστατικά απαραίτητα σε μια μηχανή αναζήτησης στο διαδίκτυο. Ο Doug Cutting οραματίστηκε το Nutch σαν ένα εναλλακτικό μέσο σε σχέση με τις εμπορικές τεχνολογίες όπως της Google.

Ακόμη, έχοντας προσθέσει συστατικά όπως ένα crawler και ένα parser, μια μηχανή αναζήτησης στο διαδίκτυο διαφέρει από μια βασική μηχανή αναζήτησης εγγράφων όσον αφορά την κλιμάκωση. Ενώ το Lucene στοχεύει στο να ευρετηριοποιήσει εκατομμύρια έγγραφα, το Nutch πρέπει να είναι ικανό να χειριστεί δισεκατομμύρια από ιστοσελίδες χωρίς να γίνεται υπερβολικά ακριβό στη λειτουργία. Το Nutch τρέχει σε ένα κατανεμημένο cluster εμπορικού hardware. Η πρόκληση για την ομάδα δημιουργίας του Nutch είναι να διευθετήσει θέματα κλιμάκωσης του λογισμικού. Το Nutch χρειάζεται ένα επίπεδο για να χειριστεί τις κατανεμημένες διεργασίες, την αφθονία, το αυτόματο failover(http://en.wikipedia.org/wiki/failover) και το load balancing(http://en.wikipedia.org/wiki/load_balancing_(computing)).Αυτές οι προκλήσεις είναι συνηθισμένες.

Γύρω στο 2004, η Google εξέδωσε δύο papers που περιέγραφαν το Google File System(GFS)(http://en.wikipedia.org/wiki/Google_File_System) και το πλαίσιο του MapReduce . Η Google ισχυρίστηκε ότι χρησιμοποίησε αυτές τις δύο τεχνολογίες για να κλιμακώσει το δικό της σύστημα αναζήτησης. Ο Doug Cutting αμέσως αντιλήφθηκε ότι μπορούσε να εφαρμόσει ιδανικά αυτές τις τεχνολογίες στο Nutch, και έτσι η ομάδα του ανέπτυξε το νέο πλαίσιο και εισήγαγε το Nutch σε αυτό. Η νέα υλοποίηση ενίσχυσε άμεσα την κλιμάκωση του Nutch. Ξεκίνησε να χειρίζεται μια σειρά από εκατοντάδες εκατομμύρια ιστοσελίδες και μπορούσε να τρέξει σε clusters από δωδεκάδες κόμβων. Ο Doug συνειδητοποίησε ότι ένα αφιερωμένο project ειδικά για να διανθίσει τις δύο τεχνολογίες ήταν επιβεβλημένο για να έχουμε κλιμάκωση στο διαδίκτυο, και έτσι το Hadoop γεννήθηκε. Η Yahoo! προσέλαβε τον Doug τον Ιανουάριο του 2006 για να εργαστεί με μια εξειδικευμένη ομάδα στην ανάπτυξη του Hadoop σαν ένα ανοιχτού κώδικα Project. Δύο χρόνια αργότερα, το Hadoop πέτυχε την κατάσταση ενός Apache Top Level Project. Λίγο αργότερα στις 19 Φεβρουαρίου του 2008 η Yahoo! ανακοίνωσε ότι το Hadoop που έτρεχε σε μια 10000+ συστάδα του πυρήνα Linux ήταν το σύστημα παραγωγής της για ευρετηριοποίηση του Web(http://developer.yahoo. net/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html). Το Hadoop είχε πραγματικά χτυπήσει τη web κλιμάκωση.

Λίγα πράγματα για τα ονόματα

Όταν ο Doug Cutting δίνει ονόματα στα project λογισμικού του εμπνέεται από την οικογένειά του. Το Lucene είναι το μεσαίο όνομα της γυναίκας του και της γιαγιάς του από τη μητέρα του το πρώτο όνομα. Ο γιος του όταν άρχισε να περπατά χρησιμοποιούσε τη λέξη Nutch σαν μια λέξη παντός σκοπού για το φαγητό και αργότερα φώναζε ένα κίτρινο ελεφαντάκι Hadoop. Όπως έχει πει "έψαχνα για ένα όνομα που δεν ήταν ήδη web domain και δεν ήταν εμπορικό σήμα,

έτσι δοκίμασα διάφορες λέξεις που υπήρχαν στη ζωή μου και δεν χρησιμοποιούνταν από άλλους. Τα παιδιά είναι αρκετά καλά στο να επινοούν λέξεις."

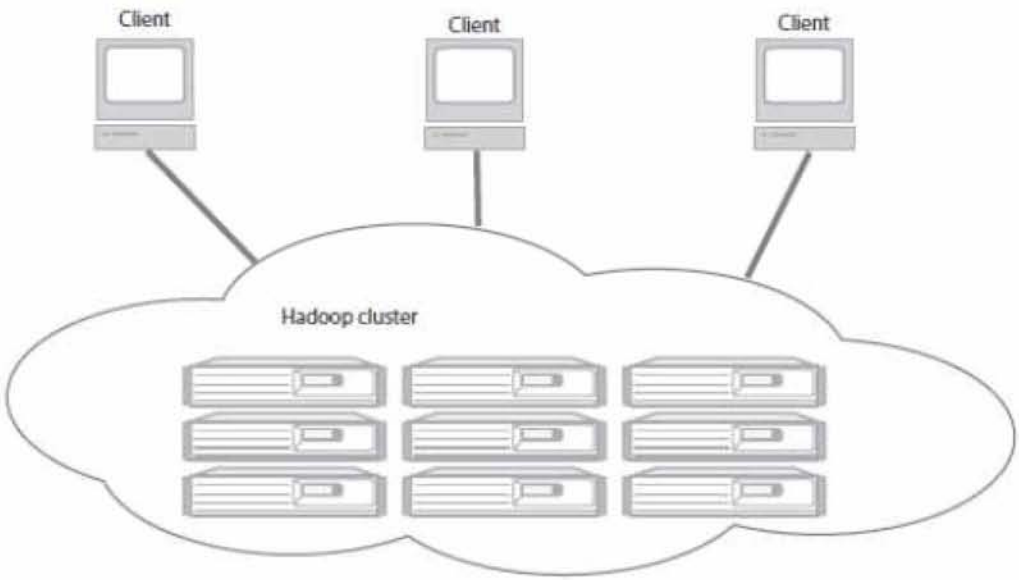
2.1Τι είναι το Hadoop

Τυπικά μιλώντας, το Hadoop είναι ένα ανοιχτού κώδικα πλαίσιο για να γράφουμε και να τρέχουμε κατανεμημένες εφαρμογές που επεξεργάζονται μεγάλες ποσότητες δεδομένων. Ο κατανεμημένος υπολογισμός είναι ένα ευρύ και ποικίλο πεδίο, αλλά οι σημαντικές διακρίσεις για το Hadoop είναι ότι είναι:

- **Προσπελάσιμο** – Το Hadoop τρέχει σε μεγάλα clusters από εμπορικές μηχανές ή σε υπηρεσίες cloud computing όπως το Elastic Compute Cloud(EC2) της Amazon.
- **Ανθεκτικό** – Γιατί προορίζεται να τρέχει σε εμπορικό υλικό, το Hadoop έχει δομηθεί με την υπόθεση των συχνών δυσλειτουργιών του υλικού. Μπορεί εύκολα να χειριστεί τέτοιες αποτυχίες.
- **Κλιμακωτό** – Το Hadoop κλιμακώνεται γραμμικά για να διαχειριστεί περισσότερα δεδομένα προσθέτοντας περισσότερους κόμβους στο cluster.
- **Απλό** – Επιτρέπει στους χρήστες να γράφουν γρήγορα αποδοτικό παράλληλο κώδικα.
- **Αξιόπιστο** – Σε περίπτωση βλάβης γίνεται αυτόματη ανάθεση των εργασιών σε νέους κόμβους. Επίσης παρέχει τη δυνατότητα αυτόματης αποθήκευσης των δεδομένων σε πολλαπλά αντίγραφα.
- **Οικονομικό** – τα δεδομένα κατανέμονται σε συστάδες αποτελούμενες από χιλιάδες κοινούς υπολογιστές

Το ότι το Hadoop είναι προσπελάσιμο και απλό βοηθάει στο να γράφουμε και τρέχουμε μεγάλα κατανεμημένα προγράμματα. Ακόμη και φοιτητές μπορούν εύκολα και οικονομικά να δημιουργήσουν το δικό τους Hadoop cluster. Από την άλλη η ανθεκτικότητα και κλιμάκωση το κάνει κατάλληλο ακόμη και για τις πιο απαιτητικές εργασίες της Yahoo και του Facebook. Αυτά τα χαρακτηριστικά καθιστούν το Hadoop δημοφιλές τόσο στην εκπαίδευση όσο και στη βιομηχανία.

Η παρακάτω εικόνα δείχνει πως κάποιος αλληλεπιδρά με ένα Hadoop cluster. Όπως μπορούμε να δούμε, ένα Hadoop cluster είναι ένα σύνολο από εμπορικές μηχανές που είναι δικτυωμένες μαζί σε μια τοποθεσία. Η αποθήκευση και επεξεργασία δεδομένων συμβαίνει μέσα στο «σύννεφο» των μηχανών. Διαφορετικοί χρήστες μπορούν να υποβάλλουν υπολογιστικές διεργασίες στο Hadoop από ξεχωριστούς πελάτες που μπορεί να είναι δικές τους επιτραπέζιες μηχανές από απομακρυσμένες τοποθεσίες από το Hadoop cluster.



Επικοινωνία Hadoop cluster και clients (Πηγή: Hadoop in Action, Chuck Lam, pg 5)

Από την άλλη πλευρά εκτός από τα πλεονεκτήματά του το Hadoop έχει και τις εξής απαιτήσεις:

- Ανάγκη γνώσης της τοπολογίας του δικτύου και χρήση αποκλειστικών switches για καλύτερο έλεγχο του bandwidth.
- Η βέλτιστη απόδοση προϋποθέτει την κατανομή των δεδομένων στους κόμβους
- Μεγάλες απαιτήσεις σε κύρια και δευτερεύουσα μνήμη.

2.2 Κατανοώντας το MapReduce

Γνωρίζουμε ήδη ορισμένα μοντέλα επεξεργασίας δεδομένων όπως τα pipelines και οι ουρές μηνυμάτων. Αυτά τα μοντέλα παρέχουν συγκεκριμένες δυνατότητες στην ανάπτυξη εφαρμογών επεξεργασίας δεδομένων.

Παρόμοια το Map Reduce είναι ένα μοντέλο επεξεργασίας δεδομένων. Το μεγαλύτερό του πλεονέκτημα είναι η ευκολία στην κλιμάκωση των δεδομένων που επεξεργάζεται όταν λειτουργεί πάνω από πολλαπλούς υπολογιστικούς κόμβους. Στο μοντέλο Map Reduce, τα θεμελιώδη συστατικά επεξεργασίας ονομάζονται mappers και reducers. Αναλύοντας μια εφαρμογή επεξεργασίας δεδομένων σε mappers και reducers συχνά είναι μη τετριμμένο. Όμως από τη στιγμή που θα γράψουμε ένα πρόγραμμα στο Map Reduce, το να κλιμακώσουμε την εφαρμογή για να τρέχει πάνω από εκατοντάδες, χιλιάδες ή δεκάδες χιλιάδες μηχανές στο cluster είναι απλά μια

αλλαγή στο configuration. Αυτή η απλή κλιμάκωση είναι ο λόγος που το μοντέλο Map Reduce προσέλκυσε πολλούς προγραμματιστές.

2.3 Κλιμακώνοντας ένα απλό πρόγραμμα χειροκίνητα

Πριν προχωρήσουμε στην τυπική χρήση του Map Reduce, ας δούμε μια άσκηση κλιμάκωσης ενός απλού προγράμματος που επεξεργάζεται ένα μεγάλο σύνολο δεδομένων. Θα δούμε τις προκλήσεις της κλιμάκωσης ενός προγράμματος επεξεργασίας δεδομένων και έτσι θα εκτιμήσουμε καλύτερα τα οφέλη της χρήσης ενός πλαισίου όπως το Map Reduce για το χειρισμό αυτής της βαρετής διαδικασίας.

Αυτό που θέλουμε να κάνουμε είναι να μετρήσουμε τον αριθμό των εμφανίσεων κάθε λέξης σε ένα σύνολο από έγγραφα. Στο συγκεκριμένο παράδειγμα έχουμε ένα σύνολο από έγγραφα που αποτελείται από ένα έγγραφο το οποίο περιέχει μόνο μια πρόταση:

Do as I say, not as I do.

Δημιουργούμε πίνακα με δύο στήλες, μια για τη λέξη και μια για τις εμφανίσεις της κάθε λέξης:

Λέξη	**Εμφανίσεις**
As	2
Do	2
I	2
not	1
Say	1

Θα ονομάζουμε αυτή την εργασία **καταμέτρηση λέξεων(word counting).** Όταν το σύνολο των εγγράφων είναι μικρό, ένα μικρό πρόγραμμα θα κάνει τη δουλειά. Ας δούμε ένα σε ψευδοκώδικα :

```
define wordCount as Multiset;
for each document in documentSet {
        T = tokenize(document);
        for each token in T {
                wordCount[token]++;
        }
}
display(wordCount);
```

Αυτό το πρόγραμμα εκτελεί ένα βρόχο περνώντας από όλα τα έγγραφα. Για κάθε έγγραφο, οι λέξεις υπολογίζονται μία προς μία κάνοντας χρήση μιας συμβολικής(tokenization) διεργασίας. Για κάθε λέξη, η αντίστοιχη είσοδος σε ένα πολυσύνολο που ονομάζεται wordCount αυξάνεται κατά ένα κάθε φορά. Τελικά μια συνάρτηση display() τυπώνει όλες καταχωρήσεις του wordCount.

Αυτό το πρόγραμμα δουλεύει καλά όσο το σύνολο των εγγράφων που θέλουμε να επεξεργαστούμε μεγαλώνει. Για παράδειγμα έστω πως θέλουμε να δημιουργήσουμε ένα φίλτρο spam μηνυμάτων έτσι ώστε να γνωρίζουμε τις πιο συχνά χρησιμοποιούμενες λέξεις στα

εκατομμύρια των spam e-mails που λαμβάνονται. Με το να εκτελούμε την ίδια διαδικασία επαναληπτικά για όλα τα έγγραφα χρησιμοποιώντας έναν και μόνο υπολογιστή θα ήταν εξαιρετικά χρονοβόρο. Μπορούμε να επιταχύνουμε ξαναγράφοντας το πρόγραμμα έτσι ώστε να κατανέμει την εργασία σε διαφορετικούς υπολογιστές. Κάθε υπολογιστής θα επεξεργάζεται ένα διαφορετικό κομμάτι των κειμένων. Όταν όλοι οι υπολογιστές ολοκληρώσουν αυτήν την επεξεργασία, μια δεύτερη φάση επεξεργασίας θα συνδυάζει τα αποτελέσματα όλων των υπολογιστών.

Ο ψευδο-κώδικας για την πρώτη φάση που θα κατανεμηθεί σε πολλούς υπολογιστές είναι

```
define wordCount as Multiset :
for each document in documentSubset {
        T = tokenize(document);
        for each token in T {
                        wordCount[token]++;
        }
}
sendToSecondPhase(wordCount);
```

Και ο ψευδο-κώδικας για την δεύτερη φάση είναι:

```
define totalWordCount as Multiset:
for each wordCount received from firstPhase {
        multisetAdd (totalWordCount, wordCount);
}
```

Παρόλο που ο παραπάνω κώδικας δεν είναι ιδιαίτερα δύσκολος, ορισμένες λεπτομέρειες μπορούν να παρεμποδίσουν την λειτουργία του. Πρώτα από όλα, αγνοούμε τις απαιτήσεις εκτέλεσης ανάγνωσης των εγγράφων. Εάν όλα τα έγγραφα είναι αποθηκευμένα σε έναν κεντρικό server αποθήκευσης, τότε η συμφόρηση θα δημιουργηθεί στο εύρος ζώνης του server. Θα πρέπει να διαχωριστούν τα έγγραφα ανάμεσα στο σύνολο των μηχανών επεξεργασίας έτσι ώστε κάθε μηχάνημα να επεξεργάζεται μόνο τόσο έγγραφα όσα είναι αποθηκευμένα σε αυτό. Αυτό θα αμβλύνει τη συμφόρηση που προκαλείται από ένα κεντρικό δίσκο αποθήκευσης. Αυτό μας υπενθυμίζει ότι τόσο η επεξεργασία όσο και η αποθήκευση πρέπει να είναι στενά συνδεδεμένες στις εφαρμογές για κατανεμημένα δεδομένα.

Ένα άλλο ελάττωμα είναι ότι το wordCount(και το totalWordCount) είναι αποθηκευμένα στη μνήμη. Όταν επεξεργαζόμαστε μεγάλα σύνολα εγγράφων, το πλήθος των λέξεων που δεν επαναλαμβάνονται μπορεί να υπερβαίνουν το μέγεθος της RAM. Για παράδειγμα η Αγγλική γλώσσα έχει γύρω στο ένα εκατομμύριο λέξεις, μέγεθος που μπορεί άνετα να αποθηκευτεί σε ένα iPod αλλά το πρόγραμμά μας θα έχει να αντιμετωπίσει μοναδικά ονόματα που δεν υπάρχουν σε κανένα αγγλικό λεξικό. Για παράδειγμα, θα πρέπει να αντιμετωπίσουμε μοναδικά ονόματα όπως το Hadoop. Επίσης θα κληθούμε να αντιμετωπίσουμε ορθογραφικά λάθη όπως τη λέξη exampel αντί της λέξης example και μετράμε όλους τους διαφορετικούς τύπους μιας λέξης ξεχωριστά(για παράδειγμα eat, ate, eaten και eating). Αλλά ακόμη και εάν ο αριθμός τον μοναδικών λέξεων στο σύνολο των εγγράφων είναι διαχειρίσιμος από τη μνήμη, μια μικρή αλλαγή στον ορισμό του προβλήματος μπορεί να εκτινάξει την πολυπλοκότητα του χώρου. Για παράδειγμα, αν αντί για λέξεις σε έγγραφα, μπορεί να θέλουμε να μετράμε τις IP διευθύνσεις σε ένα log file, ή τη

συχνότητα των bigrams(είναι ένα ζευγάρι από συνεχόμενες λέξεις). Για παράδειγμα η πρόταση "Do as I say, not as I do" μπορεί να χωριστεί στα ακόλουθα bigrams: Do as, as I, I say, say not, not as, as I, I do). Στην τελευταία περίπτωση, θα πρέπει να δουλέψουμε με ένα πολυσύνολο αποτελούμενο από εκατομμύρια εισόδους που υπερβαίνει το μέγεθος της RAM ενός εμπορικού υπολογιστή.

Το wordCount μπορεί να μην ταιριάζει στη μνήμη οπότε πρέπει να ξαναγράψουμε το πρόγραμμα έτσι ώστε να αποθηκεύει αυτόν το hash table στο δίσκο. Αυτό σημαίνει ότι πρέπει να χρησιμοποιήσουμε ένα hash table βασισμένο στο δίσκο που θα περιέχει ένα σημαντικό κομμάτι κώδικα.

Επιπλέον, θυμηθείτε ότι η φάση 2 περιλαμβάνει μόνο ένα μηχάνημα, το οποίο θα επεξεργάζεται το wordCount που στέλνεται από όλα τα μηχανήματα της φάσης 1. Το να επεξεργαζόμαστε ένα wordCount είναι από μόνο του αρκετά δύσκολο. Αφότου έχουμε προσθέσει αρκετά μηχανήματα για την επεξεργασία της φάσης 1, το μοναδικό μηχάνημα της φάσης 2 θα δημιουργήσει τη συμφόρηση. Η προφανής ερώτηση είναι : θα μπορούσαμε να ξανά γράψουμε τη φάση 2 με ένα κατανεμημένο τρόπο έτσι ώστε να κλιμακώνεται με την προσθήκη επιπλέον μηχανημάτων?

Η απάντηση είναι μπορούμε! Για να κάνουμε τη φάση 2 να λειτουργεί με κατανεμημένο τρόπο θα πρέπει κάπως να διαιρέσουμε το έργο της ανάμεσα σε πολλά μηχανήματα έτσι ώστε να μπορούν να τρέξουν ανεξάρτητα το ένα από το άλλο. Θέλουμε δηλαδή να διαιρέσουμε το wordCount μετά τη φάση 1 έτσι ώστε κάθε μηχάνημα στη φάση 2 να έχει να χειρίζεται μόνο ένα κομμάτι. Ας πούμε για παράδειγμα ότι διαθέτουμε 26 μηχανήματα για τη φάση 2. Αναθέτουμε σε κάθε μηχάνημα να χειρίζεται το wordCount για λέξεις που ξεκινούν με ένα συγκεκριμένο γράμμα της αλφαβήτου. Για παράδειγμα το μηχάνημα A στη φάση 2 θα χειρίζεται λέξεις που ξεκινούν από α. Για να κάνουμε δυνατό αυτό το διαχωρισμό στη φάση 2 χρειάζεται να προχωρήσουμε σε μια μικρή αλλαγή στη φάση 1. Αντί για ένα και μοναδικό hash table βασισμένο στο δίσκο για το wordCount, θα χρειαστούμε 26 τέτοια hash tables : wordCount-a, wordCount-b και ούτω καθεξής. Κάθε ένα μετράει λέξεις που ξεκινάνε με ένα συγκεκριμένο γράμμα. Μετά το πέρας της φάσης 1 το wordCount-a καθενός από τα μηχανήματα της φάσης 1 θα στείλει το αποτέλεσμά του στο μηχάνημα A της φάσης 2, όλα τα wordCount-b θα σταλούν στο μηχάνημα B και ούτω καθεξής. Κάθε μηχάνημα της φάσης 1 στέλνει τα αποτελέσματά του στα μηχανήματα της φάσης 2.

Γυρνώντας πίσω, αυτό το πρόγραμμα καταμέτρησης λέξεων γίνεται αρκετά πολύπλοκο. Για να το κάνουμε να δουλέψει σε ένα cluster από κατανεμημένα μηχανήματα βλέπουμε ότι πρέπει να προσθέσουμε τις ακόλουθες λειτουργικότητες:

- Αποθήκευση αρχείων σε πολλές μηχανές επεξεργασίας(στη φάση 1)
- Να γράφουμε σε ένα hash table βασισμένο στο δίσκο που θα μας επιτρέπει την επεξεργασία δεδομένων χωρίς να περιορίζεται από το μέγεθος της RAM.
- Διαχωρισμό των ενδιάμεσων δεδομένων της φάσης 1.
- Πέρασμα των διαχωρισμένων της φάσης 1 στα κατάλληλα μηχανήματα της φάσης 2.

Όλα τα παραπάνω είναι αρκετή δουλειά για κάτι τόσο απλό όσο μια καταμέτρηση λέξεων. Επίσης, δεν έχουμε ακόμη ασχοληθεί με θέματα όπως ανοχή σε σφάλματα.(Τι θα συμβεί αν ένα

μηχάνημα αποτύχει στο μέσον της εκτέλεσης?). Για όλα τα παραπάνω χρειαζόμαστε ένα framework σαν το Hadoop. Γιατί όταν γράφουμε την εφαρμογή μας στο μοντέλο Map Reduce το Hadoop φροντίζει για όλη αυτή την κλιμάκωση.

2.4 Κλιμάκωση του προβλήματος με map reduce

Τα προγράμματα που είναι γραμμένα σε MapReduce εκτελούνται σε δυο φάσεις, τη φάση του mapping και τη φάση του reducing. Κάθε φάση ορίζεται από μια συνάρτηση επεξεργασίας, mapper και reducer αντίστοιχα. Στη φάση του mapping το MapReduce λαμβάνει τα δεδομένα εισόδου και τροφοδοτεί κάθε ξεχωριστό στοιχείο τους στη συνάρτηση mapper. Στη φάση του reducing, ο reducer επεξεργάζεται όλες τις εξόδους του mapper και μας δίνει το τελικό αποτέλεσμα.

Με απλά λόγια, ο mapper φιλτράρει και μετασχηματίζει τα δεδομένα εισόδου σε κάτι που ο reducer μπορεί να συναθροίσει. Όπως φαίνεται υπάρχει μια αξιοσημείωτη ομοιότητα των δύο φάσεων του MapReduce σε σχέση με την ανάπτυξη του μετρητή λέξεων στην κλιμάκωση. Η ομοιότητα δεν είναι τυχαία. Το πλαίσιο του MapReduce σχεδιάστηκε έπειτα από χρόνια εμπειρίας σε εφαρμογές κλιμακούμενες και κατανεμημένες. Αυτό το πρότυπο των δύο φάσεων εμφανίζεται σε πολλά κλιμακωτά προγράμματα και αποτελεί βάση του πλαισίου του MapReduce.

Για να κλιμακώσουμε την κατανεμημένη εφαρμογή του μετρητή λέξεων στην προηγούμενη παράγραφο έπρεπε επίσης να γράψουμε τις συναρτήσεις partitioning και shuffling. Αυτές οι δύο συναρτήσεις είναι κοινά σχεδιαστικά πρότυπα που συνεργάζονται με τις συναρτήσεις mapping και reducing. Σε αντίθεση με τις mapping και reducing, οι partitioning και shuffling είναι γενικές λειτουργικότητες που δεν εξαρτώνται και πολύ από την εφαρμογή επεξεργασίας δεδομένων. Το πλαίσιο του MapReduce παρέχει μια εξορισμού υλοποίηση που δουλεύει στις περισσότερες περιπτώσεις.

Για να λειτουργήσουν σωστά οι mapping, reducing, partitioning και shuffling, θα πρέπει να συμφωνήσουμε σε μια κοινή δομή για την επεξεργασία των δεδομένων. Έτσι θα είναι αρκετά ευέλικτο και ισχυρό για να χειριστεί τις περισσότερες εφαρμογές στοχευμένης επεξεργασίας δεδομένων. Το MapReduce χρησιμοποιεί λίστες(lists) και ζευγάρια(key/value) σαν στοιχειώδη δεδομένα. Τα keys και values είναι συνήθως ακέραιοι ή αλφαριθμητικά καθώς επίσης μπορεί να είναι τεχνητές τιμές για να αναγνωρίζονται ή πολύπλοκοι τύποι αντικειμένων. Οι συναρτήσεις map και reduce πρέπει να υπακούουν στους ακόλουθους περιορισμούς σχετικά με τους τύπους των keys και values.

	Input	**Output**
Map	<k1,v1>	List(<k2,v2>)
Reduce	<k2,list(v2)>	List(<k3,v3>)

Στο πλαίσιο του MapReduce γράφουμε εφαρμογές καθορίζοντας τους mapper και reducer. Ας δούμε ολοκληρωμένη τη ροή δεδομένων:

1. Η είσοδος της εφαρμογής μας πρέπει να δομείται σαν μια λίστα από ζευγάρια (key/value), list(<k1,v1>). Αυτή η μορφή εισόδου μπορεί να μοιάζει αόριστη αλλά στην πράξη είναι αρκετά απλή. Η μορφή της εισόδου για την επεξεργασία πολλαπλών αρχείων είναι συνήθως της μορφής list(<String filename, String file content>). Η μορφή της εισόδου για την

επεξεργασία ενός μεγάλου αρχείου, όπως ένα log file είναι list(<Integer line_number, String log_event>).

2. Η λίστα των ζευγαριών (key/value) κατακερματίζεται και κάθε ξεχωριστό ζευγάρι(key/value), <k1,v1>, επεξεργάζεται καλώντας τη συνάρτηση map του mapper. Στην πράξη το key k1 συνήθως αγνοείται από τον mapper. Ο mapper μετατρέπει κάθε ζευγάρι <k1,v1> σε μια λίστα από ζευγάρια <k2,v2>. Οι λεπτομέρειες της μετατροπής σε μεγάλο βαθμό καθορίζουν το τι κάνει το πρόγραμμα MapReduce. Να σημειώσουμε ότι τα ζευγάρια (key/value) επεξεργάζονται με τυχαία σειρά. Η μετατροπή πρέπει να είναι αυτόνομη και λόγω αυτού η έξοδος εξαρτάται μόνο από το ζεύγος (key/value).

   Για το word counting, ο mapper παίρνει <String filename, String file_content> και γρήγορα αγνοεί το filename. Μπορεί να έχει σαν έξοδο μια λίστα από <String word, Integer count> αλλά μπορεί να είναι ακόμη πιο απλή. Όπως γνωρίζουμε το counts αθροίζεται σε ένα μεταγενέστερο στάδιο, οπότε μπορούμε να εξάγουμε μια λίστα με <String word, Integer 1> με επαναλαμβανόμενες εισόδους και να αφήσουμε το συνολικό άθροισμα για αργότερα. Έτσι στη λίστα εξόδου μπορούμε να έχουμε το ζεύγος (key/value) <"foo", 3> μια φορά ή μπορούμε να έχουμε το ζεύγος <"foo", 1> τρείς φορές. Όπως θα δούμε, η τελευταία προσέγγιση είναι περισσότερο εύκολη να προγραμματιστεί. Η προηγούμενη προσέγγιση μπορεί να έχει ορισμένα οφέλη στην απόδοση, αλλά ας αφήσουμε αυτή τη βελτιστοποίηση στην ησυχία της μέχρις ότου κατανοήσουμε πλήρως το πλαίσιο του MapReduce.

3. Η έξοδος από όλους τους mappers είναι (εννοιολογικά) άθροισμα μιας τεράστιας λίστας από ζεύγη <k2,v2>. Όλα τα ζεύγη που μοιράζονται το ίδιο k2 ομαδοποιούνται μαζί σε ένα καινούριο ζεύγος (key/value) <k2,list(v2)>. Το πλαίσιο ζητάει από το reducer να επεξεργαστεί κάθε ένα από αυτά τα σύνολα ζευγών (key/value) ξεχωριστά. Ακολουθώντας το παράδειγμα word counting, η έξοδος της συνάρτησης map για ένα έγγραφο με ένα ζεύγος <"foo",1> τρεις φορές, και η έξοδος map για άλλο έγγραφο θα είναι μια λίστα με το ζευγάρι <"foo",1> δύο φορές. Το σύνολο των ζευγών που θα παραλάβει ο reducer είναι το <"foo",list(1,1,1,1,1)>. Στον μετρητή **λέξεων,** η έξοδος του reducer θα είναι <"foo",5>, το οποίο είναι ο συνολικός αριθμός των φορών που η foo εμφανίστηκε στο σύνολο των εγγράφων. Κάθε reducer δουλεύει με μια διαφορετική λέξη. Το πλαίσιο του MapReduce αυτόματα συλλέγει όλα τα ζευγάρια <k3,v3> και τα καταγράφει σε αρχεία. Να σημειώσουμε ότι για το παράδειγμα του μετρητή **λέξεων** οι τύποι δεδομένων k2 και k3 είναι ίδιοι όπως και οι v2 και v3. Αυτό βέβαια δεν είναι δεδομένο για άλλες εφαρμογές επεξεργασίας δεδομένων.

Ας ξαναγράψουμε το πρόγραμμα του μετρητή λέξεων στο MapReduce για να δούμε πως όλα τα προηγούμενα συνδυάζονται μεταξύ τους.

```
map(String key, String value):
        // key: document name
        // value: document contents
        for each word w in value:
                EmitIntermediate(w, "1");

reduce(String key, Iterator values):
        // key: a word
        // values: a list of counts
        int result = 0;
        for each v in values:
                result += ParseInt(v);
        Emit(AsString(result));
```

Όπως αναφέρθηκε στα προηγούμενα η έξοδος για τις συναρτήσεις map και reduce είναι λίστες. Όπως μπορούμε να δούμε από τον ψευδο-κώδικα στην πράξη χρησιμοποιούμε μια ειδική συνάρτηση που ονομάζεται emit() για να γεννάμε τα στοιχεία της λίστας ένα κάθε φορά. Αυτή η συνάρτηση emit() επιπλέον απαλλάσσει τον προγραμματιστή από τη διαχείριση μιας μεγάλης λίστας.

Βιβλιογραφία

Front Page – Hadoop Wiki: http://wiki.apache.org/hadoop/

Apache Hadoop Wikipedia: http://en.wikipedia.org/wiki/Hadoop

MapReduce Wikipedia: http://en.wikipedia.org/wiki/MapReduce

Welcome to Apache Hadoop: http://hadoop.apache.org

Google Research Publication: The Google File System http://labs.google.com/papers/gfs.html

Google Research Publication: MapReduce http://labs.google.com/papers/mapreduce.html

Tom White, Hadoop: The Difinitive Guide, California: O'Reilly, 2009

Jason Venner, Pro Hadoop, New York: Springer – Verlag, 2009

Chuck Lam, Hadoop In Action, Connecticut: Manning, 2011

J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters"

3.Τρέχοντας το πρώτο μας πρόγραμμα στο Hadoop Map reduce - Μετρητής λέξεων(WordCounting)

3.1 Σκοπός

Θα περιγράψουμε τα απαιτούμενα βήματα για δημιουργία ενός **μοναδικού-κόμβου** Hadoop cluster χρησιμοποιώντας το **Hadoop Distributed File System(HDFS)** στα **Ubuntu Linux**.

Το Hadoop είναι ένα πλαίσιο γραμμένο σε Java σχεδιασμένο για τρέξιμο εφαρμογών σε μεγάλα clusters αποτελούμενα από μεγάλη ποσότητα hardware και ενσωματώνει χαρακτηριστικά όμοια με αυτά του **Google File System** και **MapReduce.** Το **HDFS** είναι ένα ανεκτικό στα λάθη κατανεμημένο σύστημα αρχείων και όπως και το Hadoop σχεδιάστηκε για να δουλεύει σε hardware χαμηλού κόστους. Παρέχει υψηλού επιπέδου πρόσβαση σε δεδομένα εφαρμογών και είναι κατάλληλο για εφαρμογές που διαθέτουν μεγάλα σύνολα δεδομένων.

Cluster από μηχανές που τρέχουν το Hadoop(Πηγή : Yahoo!)

Σκοπός μας είναι να παρουσιάσουμε μια εύκολη εγκατάσταση και λειτουργία του Hadoop έτσι ώστε να μπορέσουμε να πειραματιστούμε με το λογισμικό και να μάθουμε περισσότερα για αυτό.

Το τρέξιμο του προγράμματος έχει δοκιμαστεί με τις ακόλουθες εκδόσεις λογισμικών:

- Ubuntu Linux 11.04 που κυκλοφόρησε τον Απρίλιο του 2011 και υποστηρίζεται μέχρι τον Οκτώβριο του 2012
- Hadoop 0.20.204.0 που κυκλοφόρησε τον Αύγουστο του 2011

3.2 Απαιτήσεις Συστήματος

3.2.1 Sun Java 6

Το Hadoop προϋποθέτει μια εγκατεστημένη έκδοση της Java 1.5.x. Ωστόσο, συνίσταται η χρήση **Java 1.6.x** για το τρέξιμο Hadoop. Στο συγκεκριμένο οδηγό θα περιγράψουμε την εγκατάσταση της Java 1.6.

Στα Ubuntu **11.04**, το πακέτο sun-java6-jdk έχει παραλειφθεί. Για να το εγκαταστήσουμε πρέπει να ακολουθήσουμε τα επόμενα 4 βήματα:

1. Προσθέτουμε το Canonical Partner Repository στις apt αποθήκες μας

```
$ sudo add-apt-repository "deb http://archive.canonical.com/ lucid partner"
```

2. Αναβαθμίζουμε τη source list

```
$ sudo apt-get update
```

3. Εγκαθιστούμε το sun-java6-jdk

```
$ sudo apt-get install sun-java6-jdk
```

4. Επιλέγουμε τη Java της ως προεπιλεγμένη στο μηχάνημά μας

```
$ sudo update-java-alternatives -s java-6-sun
```

Το πλήρες JDK θα τοποθετηθεί στο **/usr/lib/jvm/java-6-sun**.

Έπειτα από την εγκατάσταση, κάντε ένα γρήγορο έλεγχο εάν το JDK έχει ρυθμιστεί σωστά:

```
user@ubuntu:~$ java -version
java version "1.6.0_26"
Java(TM) SE Runtime Environment (build 1.6.0_26-b03)
Java HotSpot(TM) Server VM (build 20.1-b02, mixed mode)
```

3.2.2 Προσθέτοντας έναν χρήστη στο Hadoop

Θα χρησιμοποιήσουμε ένα αφιερωμένο Hadoop λογαριασμό χρήστη για να τρέξουμε το Hadoop. Παρόλο που αυτό δεν απαιτείται, συνίσταται διότι βοηθάει να ξεχωρίσουμε την εγκατάσταση του Hadoop από οποιονδήποτε άλλων εφαρμογών λογισμικού και λογαριασμούς χρηστών που τρέχουν στο ίδιο μηχάνημα(για τις περιπτώσεις ασφάλειας, άδειας, backups κτλ).

```
$ sudo addgroup hadoop1
$ sudo adduser --ingroup hadoop1 hduser1
```

Οι παραπάνω εντολές θα προσθέσουν το χρήστη **hduser1** και την ομάδα **hadoop1** στο τοπικό μηχάνημά μας.

3.2.3 Διαχείριση SSH

Για τη διαχείριση των κόμβων του το Hadoop προϋποθέτει την SSH πρόσβαση, π.χ. απομακρυσμένα μηχανήματα μαζί με το τοπικό μας μηχάνημα εάν θέλουμε να χρησιμοποιούμε το Hadoop σε αυτό(που είναι και ο σκοπός αυτού του κειμένου). Για τη ρύθμιση του Hadoop ως αποτελούμενο από ένα μόνο κόμβο, θα πρέπει επίσης να ρυθμίσουμε την SSH πρόσβαση στο localhost για τον hduser χρήστη που δημιουργήσαμε στο προηγούμενο τμήμα.

Υποθέτουμε ότι έχουμε εγκαταστήσει και τρέχουμε το SSH στο μηχάνημά μας και το έχουμε τροποποιήσει έτσι ώστε να επιτρέπει SSH αυθεντικοποίηση δημόσιου κλειδιού. Εάν όχι υπάρχει μια σειρά από οδηγούς που είναι διαθέσιμοι.

Αρχικά πρέπει να δημιουργήσουμε ένα SSH κλειδί για τον **hduser** χρήστη.

```
user@ubuntu:~$ su – hduser1
hduser1@ubuntu:~$ ssh-keygen –t rsa –P “”
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser1/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Overwrite (y/n)? Y
Your identification has been saved in /home/hduser1/.ssh/id_rsa.
Your public key has been saved in /home/hduser1/.ssh/id_rsa.pub.
The key fingerprint is:
5c:86:9d:12:0c:a7:20:6e:c5:e0:2c:ea:29:46:eb:5a hduser1@ubuntu
The key's randomart image is:
```

```
+--[ RSA 2048]----+
|  ooo .oo        |
| + o.. o.+ .     |
|. =   . o =      |
|.o     . +       |
|..      S        |
|o o              |
|.=E              |
|+.               |
|o.               |
+-----------------+
hduser@ubuntu:~$
```

Η δεύτερη γραμμή θα δημιουργήσει ένα ζευγάρι κλειδιών RSA με ένα άδειο password. Γενικά η χρήση κενού password δεν συνίσταται, αλλά σε αυτή την περίπτωση είναι απαραίτητο για να αποκαλύψουμε το κλειδί χωρίς τη δική μας αλληλεπίδραση(δε θέλουμε να εισάγουμε τη φράση κλειδί κάθε φορά που το Hadoop αλληλεπιδρά με τους κόμβους του).

Δεύτερον πρέπει να ενεργοποιήσουμε την πρόσβαση του SSH στο τοπικό μας μηχάνημα με αυτό το πρόσφατα δημιουργούμενο κλειδί.

```
hduser@ubuntu:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Το τελευταίο βήμα είναι να ελέγξουμε το στήσιμο του SSH με το να συνδέσουμε το τοπικό μας μηχάνημα με τον χρήστη **hduser**. Το βήμα αυτό χρειάζεται επίσης για να σώσουμε το αποτύπωμα του host key του προσωπικού μας μηχανήματος στο αρχείο **known_hosts** του χρήστη **hduser**. Εάν έχετε ήδη κάποιο συγκεκριμένο SSH configuration για το τοπικό σας μηχάνημα όπως μια μη καθιερωμένη SSH θύρα, μπορείτε να καθορίσετε επιλογές του SSH ειδικά για host στο $HOME/.ssh/config(δείτε το man ssh_config για περισσότερες πληροφορίες)

```
hduser1@ubuntu:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 09:79:e7:9f:b1:7e:1a:ab:7d:2c:41:00:75:2c:3a:14.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Linux ubuntu 2.6.35-24-generic #42-Ubuntu SMP Thu Dec 2 01:41:57 UTC 2010 i686 GNU/Linux
Ubuntu 10.10
Welcome to Ubuntu!
 * Documentation:  https://help.ubuntu.com/

Last login: Thu Sep  8 21:39:37 2011 from localhost.localdomain
hduser1@ubuntu:~$
```

Στην περίπτωση που η σύνδεση SSH αποτύχει, οι παρακάτω συμβουλές μπορεί να βοηθήσουν:

- Ενεργοποιείστε το debugging με την εντολή **ssh –vvv localhost** και εξετάστε το λάθος λεπτομερειακά

- Ελέγξτε τη διαμόρφωση του SSH server στο **/etc/ssh/sshd_config,** συγκεκριμένα τις επιλογές **PukeyAuthentication** (το οποίο πρέπει να είναι ορισμένο στο yes) και το **AllowUsers**(εάν αυτή η επιλογή είναι ενεργή, προσθέστε τον χρήστη hduser σε αυτή). Εάν κάνετε οποιεσδήποτε αλλαγές στο SSH server configuration file, μπορείτε να εκτελέσετε μια επαναφόρτωση των ρυθμίσεων με το **sudo /etc/init.d/ssh reload**.

3.2.4 Απενεργοποίηση IPv6

Ένα πρόβλημα με το IPv6 στο Ubuntu είναι ότι χρησιμοποιώντας το **0.0.0.0** για τις διάφορες ρυθμίσεις του Hadoop που σχετίζονται με το δίκτυο έχει ως αποτέλεσμα τη δέσμευση των Ipv6 διευθύνσεων των Ubuntu. Για να απενεργοποιήσετε το IPv6 στα Ubuntu 11.04 ανοίξτε το /etc/sysctl.conf με ένα οποιδήποτε editor και προσθέστε τις ακόλουθες γραμμές στο τέλος του αρχείου:

```
#disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Στη συνέχεια κάνετε επανεκκίνηση του μηχανήματος για να λειτουργήσουν οι αλλαγές. Μπορείτε να ελένξετε εάν το IPv6 είναι ενεργοποιημένο με την ακόλουθη εντολή:

```
$ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

Εάν μας επιστρέψει τιμή 0 σημαίνει ότι το IPv6 είναι ενεργοποιημένο ενώ αν επιστρέψει τιμή 1 οτί δεν είναι ενεργοποιημένο(το οποίο είναι και αυτό

Εναλλακτικά

Μπορείτε επίσης να απενεργοποιήσετε το IPv6 μόνο για το Hadoop στο **HADOOP-3437**. Αυτό μπορεί να γίνει προσθέτοντας την ακόλουθη γραμμή στο conf/hadoop-env.sh

```
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
```

3.3 Hadoop

3.3.1 Εγκατάσταση

Αυτό που έχουμε να κάνουμε είναι να κατεβάσουμε το Hadoop από κάποιον από τους Apache Download Mirrors και να κάνουμε extract τα περιεχόμενα του πακέτου σε μια τοποθεσία της επιλογής μας. Εγώ για παράδειγμα το τοποθέτησα στο **/usr/local/hadoop**. Βεβαιωθείτε ότι έχετε αλλάξει τον ιδιοκτήτη όλων των αρχείων στο χρήστη **hduser** και στην ομάδα **hadoop**, για παράδειγμα:

```
$ cd /usr/local
$ sudo tar xzf hadoop-0.20.204.0.tar.gz
$ sudo mv hadoop-0.20.204.0 hadoop
$ sudo chown -R hduser1:hadoop1 hadoop
```

3.3.2.Αναβάθμιση $HOME/.bashrc

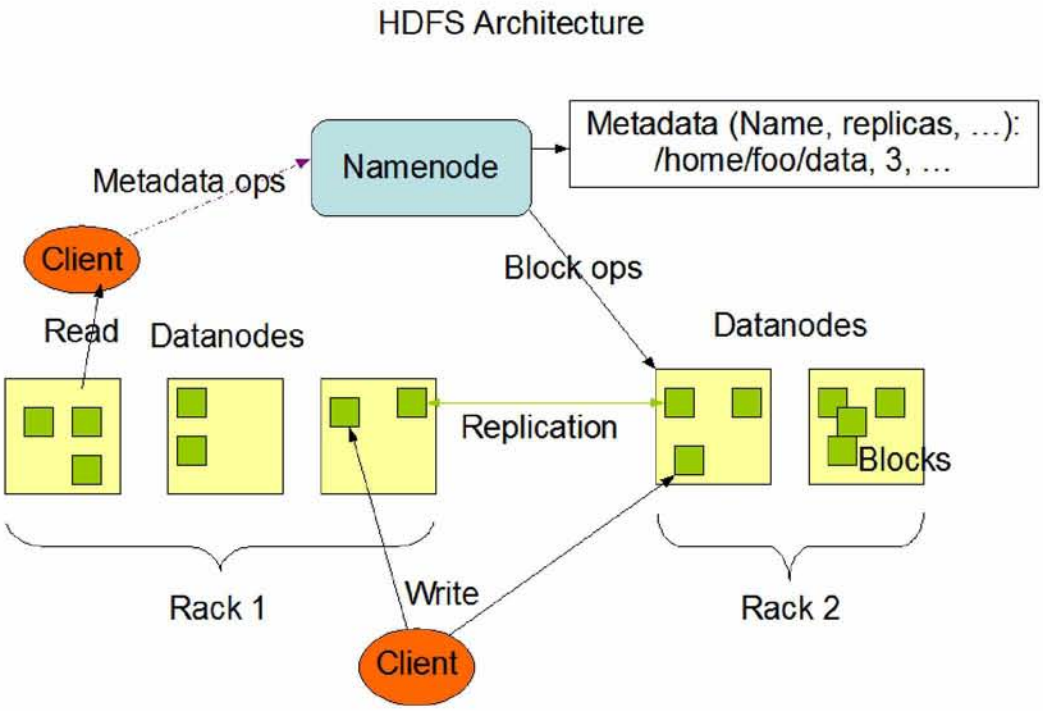
Προσθέστε τις ακόλουθες γραμμές στο τέλος του αρχείου **$HOME/.bashrc** του χρήστη **hduser**. Εάν χρησιμοποιείτε shell αντί για bash, θα πρέπει σίγουρα να αναβαθμίσετε τα κατάλληλα αρχεία τροποποίησης αντί για το **.bashrc.**

```
# Ρύθμιση των μεταβλητών που σχετίζονται με το Hadoop
export HADOOP_HOME=/usr/local/hadoop/
# Ρύθμιση του JAVA_HOME
export JAVA_HOME=/usr/lib/jvm/java-6-sun
# Ορισμένα εύκολα aliases και λειτουργίες για το τρέξιμο σχετικών # με το Hadoop εντολών
unalias fs &> /dev/null
alias fs="hadoop fs"
unalias hls &> /dev/null
alias hls="fs -ls"

# Εάν έχετε ενεργοποιήσει τη συμπίεση LZO στο cluster του Hadoop
# και η συμπίεση πετυχαίνεται με το LZOP:
# Πιο εύκολη υπόθεση μπορεί να είναι ότι ευθύνεται ένα LZOP
# συμπιεσμένο αρχείο και μπορούμε να το καταφέρουμε μέσω της
# εντολής:
#
# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
#
# Προυποθέτει την εκ των προτέρων εγκατάσταση της εντολής #'lzop'
```

```
lzohead () {
   hadoop fs -cat $1 | lzop -dc | head -1000 | less
}

# Προσθήκη του καταλόγου bin/ του Hadoop στο PATH
export PATH=$PATH:$HADOOP_HOME/bin
```

Μπορούμε να επαναλάβουμε αυτή την ίδια διαδικασία και για άλλους χρήστες που θέλουν να χρησιμοποιήσουν το Hadoop.

3.3.3.Hadoop Distributed File System (HDFS)



Η αρχιτεκτονική του HDFS (Πηγή : http://hadoop.apache.org/core/docs/current/hdfs_design.html)

3.3.3.1 Ρύθμιση Hadoop

Ο στόχος μας είναι η ρύθμιση του Hadoop για ένα μοναδικό κόμβο. Περισσότερες πληροφορίες για το τι συμβαίνει σε αυτή την ενότητα είναι διαθέσιμες στο **Hadoop Wiki**.

3.3.3.2 hadoop-env.sh

Η απαιτούμενη μεταβλητή περιβάλλοντος που πρέπει να διαχειριστούμε για το Hadoop είναι η **JAVA_HOME**. Ανοίξτε το /conf/hadoop-env.sh σε ένα editor της επιλογής σας(εάν χρησιμοποιήσατε το δικό μας path εγκατάστασης, το πλήρες path είναι **/usr/local/hadoop/conf/hadoop-env.sh**) και ορίστε τη μεταβλητή περιβάλλοντος **JAVA_HOME** στον κατάλογο του JDK/JRE 6.

Αλλάξτε το

```
# The java implementation to use.  Required.
# export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

στο

```
# The java implementation to use.  Required.
export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

3.3.3.3 conf/*-site.xml

Σε αυτό το τμήμα, θα χειριστούμε τον κατάλογο όπου το Hadoop θα αποθηκεύει τα αρχεία δεδομένων του, τη θύρα του δικτύου που αυτό ακούει, κτλ. Η εγκατάστασή μας θα χρησιμοποιήσει το Hadoop's Distributed File System, **HDFS**, ακόμα και εάν το μικρό μας cluster περιέχει μόνο το μοναδικό τοπικό μας μηχάνημα.

Μπορούμε να αφήσουμε τις πιο πάνω ρυθμίσεις όπως έχουν, με εξαίρεση τη μεταβλητή hadoop.tmp.dir όπου έχουμε να τη μεταφέρουμε σε κατάλογο της επιλογής μας. Θα χρησιμοποιήσουμε τον κατάλογο **/app/hadoop/tmp**. Οι default ρυθμίσεις του Hadoop χρησιμοποιούν το **hadoop.tmp.dir** σαν το βασικό προσωρινό κατάλογο τόσο για το τοπικό file system όσο και στο HDFS. Έτσι μην εκπλήσεστε αν δείτε το Hadoop να δημιουργεί τον εξειδικευμένο κατάλογο αυτόματα στο HDFS σε κάποιο αργότερο σημείο.

Τώρα θα δημιουργήσουμε τον κατάλογο και θα ορίσουμε τις απαιτούμενες ιδιοκτησίες και άδειες:

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser1:hadoop1 /app/hadoop/tmp
# εάν θέλετε μεγαλύτερη ασφάλεια αλλάξτε το chmod από 755 σε # 750...
$ sudo chmod 750 /app/hadoop/tmp
```

Εάν ξεχάσετε να θέσετε τις απαιτούμενες ιδιοκτησίες και άδειες, θα δείτε ένα java.io.IOException όταν θα προσπαθήσετε να κάνετε format το όνομα του κόμβου στην επόμενη παράγραφο.

Προσθέστε τα ακόλουθα μικρά τμήματα ανάμεσα στα tags **<configuration>…</configuration>** στο αντίστοιχο XML αρχείο διαμόρφωσης.

Στο αρχείο **conf/core-site.xml:**

```
<!-- In: conf/core-site.xml -->
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
</property>

<property>
  <name>fs.default.name</name>
  <value>hdfs://localhost:54310</value>
  <description>The name of the default file system.  A URI whose
  scheme and authority determine the FileSystem implementation.  The
  uri's scheme determines the config property (fs.SCHEME.impl) naming
  the FileSystem implementation class.  The uri's authority is used to
  determine the host, port, etc. for a filesystem.</description>
</property>
```

Στο αρχείο **conf/mapred-site.xml:**

```
<!-- In: conf/mapred-site.xml -->
<property>
  <name>mapred.job.tracker</name>
  <value>localhost:54311</value>
  <description>The host and port that the MapReduce job tracker runs
  at.  If "local", then jobs are run in-process as a single map
  and reduce task.
  </description>
</property>
```

Στο αρχείο **conf/hdfs-site.xml:**

```
<!-- In: conf/hdfs-site.xml -->
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.
  The actual number of replications can be specified when the file is created.
  The default is used if replication is not specified in create time.
  </description>
```

```
</property>
```

Δείτε το **Getting Started with Hadoop** και την τεκμηρίωση στο **Hadoop's API Overview** εάν έχετε επιπλέον ερωτήσεις σχετικά με τις επιλογές διαμόρφωσης του Hadoop.

3.3.5 Διαμορφώνοντας τον name node

Το πρώτο βήμα για να ξεκινήσουμε την εγκατάσταση του Hadoop είναι να κάνουμε format το Hadoop filesystem το οποίο εκτελείται στην κορυφή του τοπικού filesystem του δικού μας cluster(το οποίο περιλαμβάνει μόνο το τοπικό μας μηχάνημα). Αυτό πρέπει να γίνει την πρώτη φορά που ρυθμίζουμε το Hadoop cluster.

Μην κάνετε format ένα Hadoop filesystem που τρέχει γιατί θα χάσετε όλα τα δεδομένα που βρίσκονται εκείνη τη στιγμή στο cluster(στο HDFS).

Για να κάνετε format το filesystem (που απλά αρχικοποιεί τον κατάλογο που καθορίζεται από τη μεταβλητή dfs.name.dir), τρέξτε την εντολή:

```
hduser1@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode -format
```

Η έξοδος θα είναι ως εξής:

```
hduser1@ubuntu:/usr/local/hadoop$ /bin/hadoop namenode –format
11/09/09 00:19:25 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ubuntu/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 0.20.204.0
STARTUP_MSG:   build = http://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20-
security-203 -r 1099333; compiled by 'oom' on Wed May  4 07:57:50 PDT 2011
************************************************************/
Re-format filesystem in /app/hadoop/tmp/dfs/name ? (Y or N) Y
11/09/09 00:20:53 INFO util.GSet: VM type       = 32-bit
11/09/09 00:20:53 INFO util.GSet: 2% max memory = 17.77875 MB
11/09/09 00:20:53 INFO util.GSet: capacity      = 2^22 = 4194304 entries
11/09/09 00:20:53 INFO util.GSet: recommended=4194304, actual=4194304
11/09/09 00:20:53 INFO namenode.FSNamesystem: fsOwner=hduser
11/09/09 00:20:53 INFO namenode.FSNamesystem: supergroup=supergroup
11/09/09 00:20:53 INFO namenode.FSNamesystem: isPermissionEnabled=true
11/09/09 00:20:53 INFO namenode.FSNamesystem: dfs.block.invalidate.limit=100
11/09/09 00:20:53 INFO namenode.FSNamesystem: isAccessTokenEnabled=false
accessKeyUpdateInterval=0 min(s), accessTokenLifetime=0 min(s)
11/09/09 00:20:53 INFO namenode.NameNode: Caching file names occuring more than 10 times
```

```
11/09/09 00:20:54 INFO common.Storage: Image file of size 112 saved in 0 seconds.
11/09/09 00:20:54 INFO common.Storage: Storage directory /app/hadoop/tmp/dfs/name has been
successfully formatted.
11/09/09 00:20:54 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at ubuntu/127.0.1.1
************************************************************/
hduser1@ubuntu:/usr/local/hadoop$
```

3.3.6 Εκκίνηση ενός single-node cluster

Τρέξτε την εντολή:

```
hduser1@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```

Αυτή η εντολή θα εκκινήσει ένα Namenode, Datanode, Jobtracker και ένα Tasktracker στο μηχάνημά μας.

Η έξοδος θα μοιάζει όπως παρακάτω:

```
hduser1@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
starting namenode, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser1-
namenode-ubuntu.out
localhost: starting datanode, logging to /usr/local/hadoop/libexec/../logs/hadoop-
hduser1-datanode-ubuntu.out
localhost: starting secondarynamenode, logging to
/usr/local/hadoop/libexec/../logs/hadoop-hduser1-secondarynamenode-ubuntu.out
starting jobtracker, logging to /usr/local/hadoop/libexec/../logs/hadoop-hduser1-
jobtracker-ubuntu.out
localhost: starting tasktracker, logging to /usr/local/hadoop/libexec/../logs/hadoop-
hduser1-tasktracker-ubuntu.out
hduser1@ubuntu:~$
```

Ένα πολύ καλό εργαλείο για το έλεγχο του εάν τρέχουν οι αναμενόμενες Hadoop διεργασίες είναι το jps(μέρος της Java της Sun από την έκδοση v1.5.0). Δείτε επίσης το **How to debug MapReduce programs.**

```
hduser1@ubuntu:/usr/local/hadoop$ jps
16694 Jps
16341 JobTracker
15857 NameNode
16046 DataNode
16530 TaskTracker
16245 SecondaryNameNode
```

Μπορείτε επίσης να ελένξετε με το **netstat** εάν το Hadoop ακούει στις

διαμορφωμένες θύρες.

```
hduser1@ubuntu:~$ sudo netstat -plten | grep java
tcp        0      0 0.0.0.0:56944           0.0.0.0:*               LISTEN      1001       79140
6750/java
tcp        0      0 0.0.0.0:50070           0.0.0.0:*               LISTEN      1001       80232
6750/java
tcp        0      0 0.0.0.0:59418           0.0.0.0:*               LISTEN      1001       79824
7097/java
tcp        0      0 127.0.0.1:52005         0.0.0.0:*               LISTEN      1001       80654
7336/java
tcp        0      0 127.0.0.1:54310         0.0.0.0:*               LISTEN      1001       79402
6750/java
tcp        0      0 127.0.0.1:54311         0.0.0.0:*               LISTEN      1001       80168
7169/java
tcp        0      0 0.0.0.0:50090           0.0.0.0:*               LISTEN      1001       80254
7097/java
tcp        0      0 0.0.0.0:33611           0.0.0.0:*               LISTEN      1001       79930
7169/java
tcp        0      0 0.0.0.0:50030           0.0.0.0:*               LISTEN      1001       80331
7169/java
hduser1@ubuntu:~$
```

Εάν υπάρχουν σφάλματα, εξετάστε τα log files στον κατάλογο /logs/.

3.3.7 Σταμάτημα ενός single-node cluster

Τρέξτε την εντολή

```
hduser1@ubuntu:~$ /usr/local/hadoop/bin/stop-all.sh
```

για να σταματήσετε όλα τα daemons που τρέχουν στο μηχάνημά σας.

Ένα υπόδειγμα εξόδου είναι το παρακάτω:

```
hduser1@ubuntu:/usr/local/hadoop$ bin/stop-all.sh
stopping jobtracker
localhost: stopping tasktracker
stopping namenode
localhost: stopping datanode
localhost: stopping secondarynamenode
hduser1@ubuntu:/usr/local/hadoop$
```

3.3.8 Τρέχοντας μια διεργασία στο MapReduce

Τώρα θα τρέξουμε την πρώτη μας Hadoop MapReduce εργασία. Θα χρησιμοποιήσουμε την **WordCount example job** η οποία διαβάζει αρχεία κειμένου και μετράει τη συχνότητα εμφάνισης των λέξεων. Η είσοδος είναι αρχεία κειμένου και η έξοδος είναι επίσης αρχεία κειμένου, κάθε γραμμή των οποίων περιέχει μια λέξη και την καταμέτρηση του πόσο συχνά αυτή εμφανίζεται, διαχωρισμένα με ένα στηλοθέτη. Περισσότερες πληροφορίες για το τι συμβαίνει στο παρασκήνιο είναι διαθέσιμες στο Hadoop Wiki.

3.3.8.1 Κατέβασμα των δεδομένων εισόδου του παραδείγματος

Θα χρησιμοποιήσουμε τρία βιβλία από το Gutenberg Project για αυτό το παράδειγμα:

- The Outline of Science, Vol. 1 (of 4) by J. Arthur Thomson
- The Notebooks of Leonardo Da Vinci
- Ulysses by James Joyce

Κατεβάστε κάθε βιβλίο σαν αρχεία text σε **Plain Text UTF-8** encoding και αποθηκεύστε τα αρχεία σε ένα προσωρινό κατάλογο, για παράδειγμα στον **/tmp/gutenberg**.

```
hduser1@ubuntu:~$ ls -l /tmp/gutenberg/
total 3592
-rw-r--r-- 1 dimitris dimitris  674566 2011-09-09 01:19 pg20417.txt
-rw-r--r-- 1 dimitris dimitris 1573112 2011-09-09 01:20 pg4300.txt
-rw-r--r-- 1 dimitris dimitris 1423801 2011-09-09 01:19 pg5000.txt
hduser1@ubuntu:~$
```

3.3.8.2 Επανεκκίνηση του Hadoop cluster

Επανεκκινήστε τον Hadoop cluster εάν δεν τρέχει ήδη.

```
hduser1@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```

3.3.8.3 Αντιγραφή των δεδομένων στο HDFS

Πριν τρέξουμε την πραγματική εργασία MapReduce, πρώτα αντιγράφουμε τα αρχεία από το τοπικό μας file system στο **HDFS** του Hadoop.

```
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -copyFromLocal /tmp/gutenberg/
/usr/hduser/gutenberg
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /usr/hduser
Found 1 items
drwxr-xr-x   - hduser1 supergroup          0 2011-09-10 00:31 /usr/hduser/gutenberg
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /usr/hduser/gutenberg
Found 3 items
-rw-r--r--   1 hduser1 supergroup     674566 2011-09-10 00:31 /usr/hduser/gutenberg/pg20417.txt
-rw-r--r--   1 hduser1 supergroup    1573112 2011-09-10 00:31 /usr/hduser/gutenberg/pg4300.txt
-rw-r--r--   1 hduser1 supergroup    1423801 2011-09-10 00:31 /usr/hduser/gutenberg/pg5000.txt
hduser1@ubuntu:/usr/local/hadoop$
```

3.3.8.4 Τρέξιμο στο MapReduce

Τώρα στην πραγματικότητα τρέχουμε το παράδειγμα της εργασίας του WordCount.

```
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop jar hadoop-examples-0.20.204.0.jar wordcount
/usr/hduser/gutenberg /usr/hduser/gutenberg-output
```

Αυτή η εντολή θα διαβάσει όλα τα αρχεία που βρίσκονται στον κατάλογο του HDFS **/usr/hduser/gutenberg**, θα τα επεξεργαστεί και θα αποθηκεύσει το αποτέλεσμα στον κατάλογο του HDFS **/usr/hduser/gutenberg-output.**

Ένα υπόδειγμα εξόδου της προηγούμενης εντολής στην κονσόλα είναι:

```
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop jar hadoop-examples-0.20.204.0.jar wordcount
/usr/hduser/gutenberg /usr/hduser/gutenberg-output
11/09/10 00:39:35 INFO input.FileInputFormat: Total input paths to process : 3
11/09/10 00:39:36 INFO mapred.JobClient: Running job: job_201109100022_0001
11/09/10 00:39:37 INFO mapred.JobClient:  map 0% reduce 0%
11/09/10 00:39:55 INFO mapred.JobClient:  map 66% reduce 0%
11/09/10 00:40:01 INFO mapred.JobClient:  map 100% reduce 0%
11/09/10 00:40:04 INFO mapred.JobClient:  map 100% reduce 33%
11/09/10 00:40:10 INFO mapred.JobClient:  map 100% reduce 100%
11/09/10 00:40:15 INFO mapred.JobClient: Job complete: job_201109100022_0001
```

```
11/09/10 00:40:15 INFO mapred.JobClient: Counters: 25
11/09/10 00:40:15 INFO mapred.JobClient:   Job Counters
11/09/10 00:40:15 INFO mapred.JobClient:     Launched reduce tasks=1
11/09/10 00:40:15 INFO mapred.JobClient:     SLOTS_MILLIS_MAPS=31900
11/09/10 00:40:15 INFO mapred.JobClient:     Total time spent by all reduces waiting after
reserving slots (ms)=0
11/09/10 00:40:15 INFO mapred.JobClient:     Total time spent by all maps waiting after reserving
slots (ms)=0
11/09/10 00:40:15 INFO mapred.JobClient:     Launched map tasks=3
11/09/10 00:40:15 INFO mapred.JobClient:     Data-local map tasks=3
11/09/10 00:40:15 INFO mapred.JobClient:     SLOTS_MILLIS_REDUCES=14642
11/09/10 00:40:15 INFO mapred.JobClient:   File Output Format Counters
11/09/10 00:40:15 INFO mapred.JobClient:     Bytes Written=880802
11/09/10 00:40:15 INFO mapred.JobClient:   FileSystemCounters
11/09/10 00:40:15 INFO mapred.JobClient:     FILE_BYTES_READ=2214725
11/09/10 00:40:15 INFO mapred.JobClient:     HDFS_BYTES_READ=3671837
11/09/10 00:40:15 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=3773669
11/09/10 00:40:15 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=880802
11/09/10 00:40:15 INFO mapred.JobClient:   File Input Format Counters
11/09/10 00:40:15 INFO mapred.JobClient:     Bytes Read=3671479
11/09/10 00:40:15 INFO mapred.JobClient:   Map-Reduce Framework
11/09/10 00:40:15 INFO mapred.JobClient:     Reduce input groups=82331
11/09/10 00:40:15 INFO mapred.JobClient:     Map output materialized bytes=1474279
11/09/10 00:40:15 INFO mapred.JobClient:     Combine output records=102317
11/09/10 00:40:15 INFO mapred.JobClient:     Map input records=77931
11/09/10 00:40:15 INFO mapred.JobClient:     Reduce shuffle bytes=1474279
11/09/10 00:40:15 INFO mapred.JobClient:     Reduce output records=82331
11/09/10 00:40:15 INFO mapred.JobClient:     Spilled Records=255947
11/09/10 00:40:15 INFO mapred.JobClient:     Map output bytes=6076039
11/09/10 00:40:15 INFO mapred.JobClient:     Combine input records=629167
11/09/10 00:40:15 INFO mapred.JobClient:     Map output records=629167
11/09/10 00:40:15 INFO mapred.JobClient:     SPLIT_RAW_BYTES=358
11/09/10 00:40:15 INFO mapred.JobClient:     Reduce input records=102317
hduser1@ubuntu:/usr/local/hadoop$
```

Ελένξτε εάν το αποτέλεσμα αποθηκεύτηκε επιτυχώς στον κατάλογο του HDFS **/usr/hduser/gutenberg-output**:

```
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /usr/hduser
Found 2 items
drwxr-xr-x   - hduser1 supergroup          0 2011-09-10 00:31 /usr/hduser/gutenberg
drwxr-xr-x   - hduser1 supergroup          0 2011-09-10 00:40 /usr/hduser/gutenberg-output
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -ls /usr/hduser/gutenberg-output
Found 3 items
-rw-r--r--   1 hduser1 supergroup          0 2011-09-10 00:40 /usr/hduser/gutenberg-
output/_SUCCESS
drwxr-xr-x   - hduser1 supergroup          0 2011-09-10 00:39 /usr/hduser/gutenberg-output/_logs
-rw-r--r--   1 hduser1 supergroup     880802 2011-09-10 00:40 /usr/hduser/gutenberg-output/part-r-
00000
```

```
hduser1@ubuntu:/usr/local/hadoop$
```

Εάν θέλετε να τροποποιήσετε ορισμένες από τις ρυθμίσεις του Hadoop κατά τη διάρκεια της εκτέλεσης, όπως για παράδειγμα να αυξήσετε τον αριθμό των διεργασιών Reduce, μπορείτε να χρησιμοποιήσετε την επιλογή "-D":

```
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop jar hadoop*examples*.jar wordcount -D
mapred.reduce.tasks=16 /usr/hduser/gutenberg /usr/hduser/gutenberg-output
```

3.3.8.5 Εξόρυξη αποτελεσμάτων από το HDFS

Για να εξετάσουμε το αρχείο μπορούμε να το αντιγράψουμε από το HDFS στο τοπικό file system. Εναλλακτικά μπορούμε να χρησιμοποιήσουμε την εντολή

```
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -cat /user/hduser/gutenberg-
output/part-r-00000
```

για να διαβάσουμε το αρχείο απευθείας από το HDFS χωρίς να το αντιγράψουμε στο τοπικό file system. Στη δική μας περίπτωση θα αντιγράψουμε τα αποτελέσματα στο τοπικό file system.

```
hduser1@ubuntu:/usr/local/hadoop$ mkdir /tmp/gutenberg-output
hduser1@ubuntu:/usr/local/hadoop$ bin/hadoop dfs -getmerge /usr/hduser/gutenberg-output
/tmp/gutenberg-output
hduser1@ubuntu:/usr/local/hadoop$ head /tmp/gutenberg-output/gutenberg-output
"(Lo)cra"       1
"1490   1
"1498,"         1
"35"    1
"40,"   1
"A      2
"AS-IS".        1
"A_     1
"Absoluti       1
"Alack!         1
hduser1@ubuntu:/usr/local/hadoop$
```

Να σημειώσουμε ότι στη συγκεκριμένη έξοδο τα εισαγωγικά (") δεν έχουν εισαχθεί από το Hadoop. Είναι το αποτέλεσμα από το word tokenizer που χρησιμοποιήθηκε από το παράδειγμα WordCount και σε αυτή την περίπτωση ταίριαξαν το ξεκίνημα ενός εισαγωγικού στα κείμενα του βιβλίου. Απλά εξετάστε το αρχείο **part-00000** περισσότερο για να το διαπιστώσετε.

3.4 Hadoop Web Interfaces

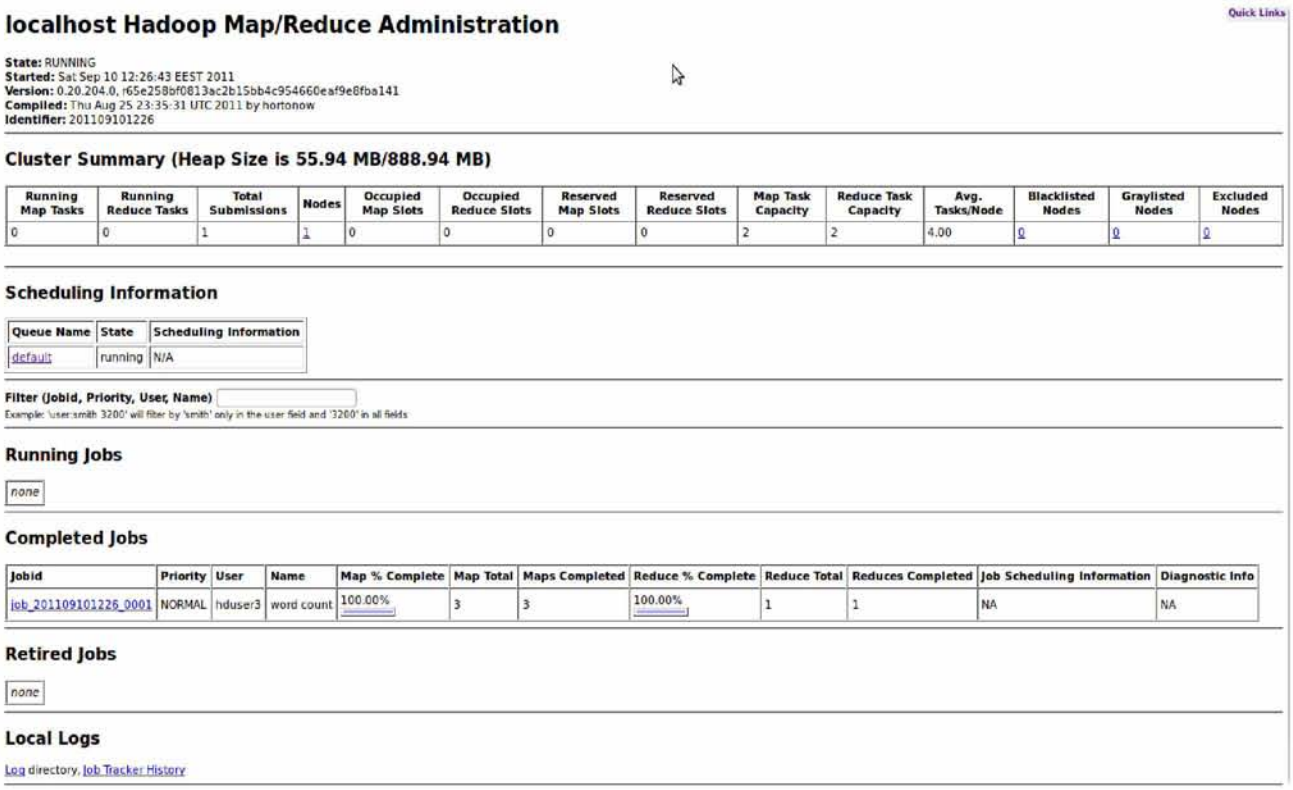
Το Hadoop συνοδεύεται από αρκετά web interfaces τα οποία είναι εξορισμού (δείτε το **conf/hadoop-default.xml**) διαθέσιμα στις κάτωθι τοποθεσίες :

- http://localhost:50030/ – web UI for MapReduce job tracker(s)
- http://localhost:50060/ – web UI for task tracker(s)
- http://localhost:50070/ – web UI for HDFS name node(s)

Αυτά τα web interfaces παρέχουν σύντομες πληροφορίες για το τι συμβαίνει στο Hadoop cluster μας. Θα ήταν καλό να τα δοκιμάσετε.

3.4.1 MapReduce Job Tracker Web Interface

Ο job tracker web UI παρέχει πληροφορίες για γενικά στατιστικά των εργασιών του Hadoop cluster, τρέχοντες/ολοκληρωμένες/αποτυχημένες εργασίες και ένα αρχείο log για το ιστορικό. Αυτό επίσης δίνει πρόσβαση στα log files του Hadoop στο τοπικό μηχάνημα(το μηχάνημα στο οποίο το web UI τρέχει επάνω)
Εξορισμού είναι διαθέσιμο στο **http://localhost:50030/**.



localhost Hadoop Map/Reduce Administration

Quick Links

State: RUNNING
Started: Sat Sep 10 12:26:43 EEST 2011
Version: 0.20.204.0, r65e258bf0813ac2b15bb4c954660eaf9e8fba141
Compiled: Thu Aug 25 23:35:31 UTC 2011 by hortonow
Identifier: 201109101226

Cluster Summary (Heap Size is 55.94 MB/888.94 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	1	1	0	0	0	0	2	2	4.00	0	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

none

Completed Jobs

Jobid	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201109101226_0001	NORMAL	hduser3	word count	100.00%	3	3	100.00%	1	1	NA	NA

Retired Jobs

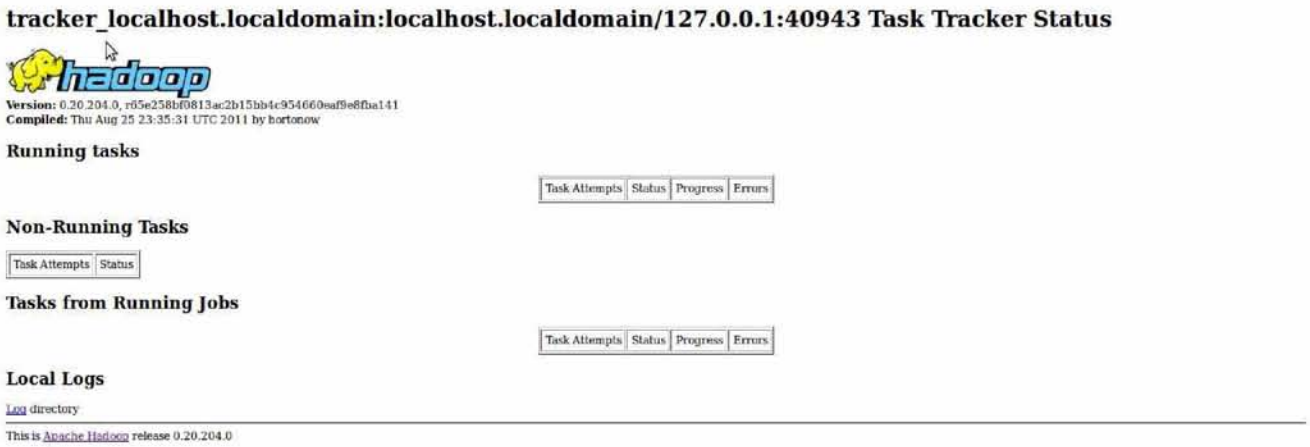
none

Local Logs

Log directory, Job Tracker History

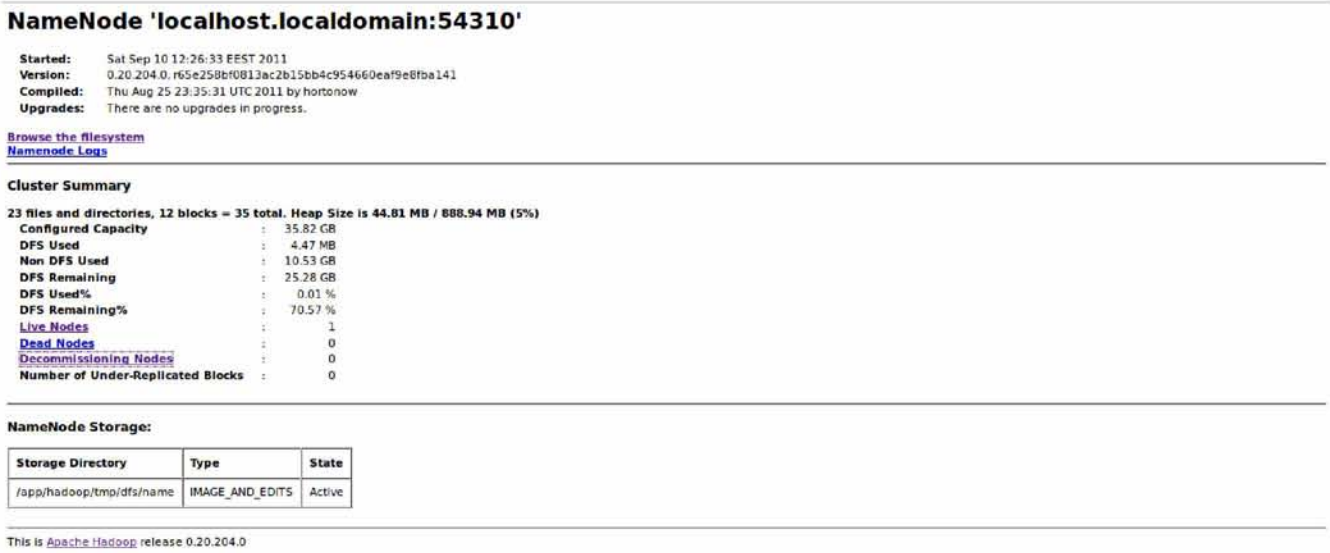
3.4.2 Task Tracker Web Interface

Ο task tracker web UI μας δείχνει τις εργασίες που τρέχουν και αυτές που δεν τρέχουν. Επίσης μας δίνει πρόσβαση στα log files του Hadoop στο τοπικό μηχάνημα.
Εξορισμού είναι διαθέσιμο στο http://localhost:50060/.



3.4.3 HDFS Name Node Web Interface

Ο name node web UI περιέχει μια σύνοψη του cluster περιλαμβάνοντας πληροφορίες για τη συνολική/εναπομένουσα χωρητικότητα, ανενεργούς και ενεργούς κόμβους. Επιπρόσθετα, επιτρέπει να φυλοομετρήσουμε το HDFS namespace και να δούμε τα περιεχόμενα των αρχείων του στον web browser. Επίσης δίνει πρόσβαση στα log files του Hadoop στο τοπικό μηχάνημα.
Εξορισμού είναι διαθέσιμο στο http://localhost:50070/.



NameNode 'localhost.localdomain:54310'

Started: Sat Sep 10 12:26:33 EEST 2011
Version: 0.20.204.0, r65e258bf0813ac2b15bb4c954660eaf9e8fba141
Compiled: Thu Aug 25 23:35:31 UTC 2011 by hortonow
Upgrades: There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

23 files and directories, 12 blocks = 35 total. Heap Size is 44.81 MB / 888.94 MB (5%)

Configured Capacity	:	35.82 GB
DFS Used	:	4.47 MB
Non DFS Used	:	10.53 GB
DFS Remaining	:	25.28 GB
DFS Used%	:	0.01 %
DFS Remaining%	:	70.57 %
Live Nodes	:	1
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0

NameNode Storage:

Storage Directory	Type	State
/app/hadoop/tmp/dfs/name	IMAGE_AND_EDITS	Active

This is Apache Hadoop release 0.20.204.0

Βιβλιογραφία

Welcome to Apache Hadoop: http://hadoop.apache.org

Getting Started with Hadoop http://wiki.apache.org/hadoop/GettingStartedWithHadoop

Project Description http://wiki.apache.org/hadoop/ProjectDescription

How to debug MapReduce programs
http://wiki.apache.org/hadoop/HowToDebugMapReducePrograms

Hadoop API Overview http://hadoop.apache.org/common/docs/current/api/overview-summary.html

Running Hadoop On Ubuntu Linux(Single-Node Cluster), Michael G. Noll, http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

The Outline of Science, Vol. 1 (of 4) by J. Arthur Thomson :
http://www.gutenberg.org/ebooks/20417
The Notebooks of Leonardo Da Vinci : http://www.gutenberg.org/ebooks/5000
Ulysses by James Joyce : http://www.gutenberg.org/ebooks/4300

ΜΕΡΟΣ Β

4. Βελτίωση της αποδοτικότητας μιας διαφήμισης

Μια μηχανή αναζήτησης αποφασίζει ποιες διαφημίσεις θα δείξει (και με ποια σειρά), με κύριο σκοπό να αυξήσει τα έσοδα σε αντιδιαστολή με το κόστος, στηριζόμενη στην πιθανότητα μια διαφήμιση να επιλεγεί[4]. Εκτός από την επιλογή και την ιεράρχηση των διαφημίσεων, πρέπει παράλληλα να αποφασίζει πόσες διαφημίσεις θα δείξει και για πόσο χρονικό διάστημα. Έτσι ενώ μπορεί να αυξήσει τα έσοδα βραχυπρόθεσμα (αυξάνοντας τον αριθμό των αποτελεσμάτων), μια τέτοια προσέγγιση θα μείωνε την ποιότητα και το αποτέλεσμα θα ήταν οι χρήστες να χρησιμοποιήσουν μια εναλλακτική επιλογή. Κάθε μηχανή αναζήτησης διαλέγει το πόσο επιθετικά θα διαφημίσει ισορροπώντας ανάμεσα στους επιχειρηματικούς στόχους της αύξησης των εσόδων και στην επίδραση(επιρροή) που έχει στον χρήστη η εκάστοτε πολιτική διαφήμισης. Τα αποτελέσματα που παρέχουν μεγάλες εμπορικές μηχανές επιλέγονται από μια βάση δεδομένων που αποτελείται από διαφημιστές που προσφέρουν χρήματα (bid) για να έχουν την διαφήμισή τους στην σελίδα αναζήτησης.

Η συγκεκριμένη εργασία εστιάζει στην πρόβλεψη της συνάφειας των διαφημίσεων(σε σχέση με το ερώτημα αναζήτησης), έτσι ώστε αυτόματα να καταλαβαίνουμε τις διαφημίσεις χαμηλής σχετικότητας. Ακόμη ενισχύει την ικανότητά βελτιστοποίησης και αξιολόγησης του συστήματος αναζήτησης.

Πιο συγκεκριμένα, μοντελοποιεί την συνάφεια των διαφημίσεων(ads) στην προσπάθεια βελτιστοποίησης του συστήματος αναζήτησης. Παρόλο που η συνάφεια και τα κλικ σχετίζονται αρκετά μεταξύ τους, εν τούτοις παρουσιάζουν σημαντικές διαφορές. Η αξιολόγηση άρθρων με πολλά κοινά στοιχεία, μας δείχνει πόσο σχετική είναι μια διαφήμιση με ένα ερώτημα αναζήτησης (search query) ενώ ο αριθμός εμφανίσεων(CTR) δείχνει το βαθμό που μια διαφήμιση είναι ελκυστική. Αυτά τα δυο μέτρα(συνάφεια και κλικς) μπορεί να αποκλίνουν. Μια διαφήμιση π.χ. "αγόρασε coca cola online" σχετίζεται πολύ με την αναζήτηση coca cola, αλλά το CTR(κόστος ανά κλικ) είναι χαμηλό, επειδή πολλοί λίγοι άνθρωποι επιλέγουν να αγοράσουν coca cola από το internet. Από την άλλη, είναι ελκυστική η εύρεση εργασίας(από το internet) στην coca cola για έναν χρήστη. Συνεπώς μπορεί η αναζήτηση δουλειάς στην coca cola είναι λιγότερο σχετική με την ερώτηση("αγόρασε coca cola online"), αλλά έχει μεγαλύτερο CTR επειδή η διαφήμιση είναι πολύ ελκυστική στους χρήστες.

4.1 Ανασκόπιση αναζήτησης χρηματοδοτούμενων διαφημίσεων

Έπειτα από κάθε αναζήτηση του χρήστη, οι μηχανές αναζήτησης εκθέτουν στην οθόνη(πάνω - δεξιά) τις διαφημίσεις οι οποίες χρηματοδοτούνται από τους διαφημιστές. Το μοντέλο εσόδων για αυτές τις λίστες είναι το CTR(click through rate), όπου ο διαφημιζόμενος πληρώνει μόνο αν η διαφήμιση επιλέγεται. Ο διαφημιστής στοχεύει συγκεκριμένες αγορές με λέξεις κλειδιά, προσφέροντας χρήματα για τις συγκεκριμένες αναζητήσεις. Για παράδειγμα ένας

διαφημιστής που πουλάει παπούτσια μπορεί να προσφέρει λεφτά σε αναζητήσεις όπως "φθηνά παπούτσια" ή "παπούτσια για τρέξιμο". Η χρηματοδοτούμενη αναζήτηση προσφέρει πιο στοχευμένο και λιγότερο ακριβό τρόπο marketing για τους περισσότερους διαφημιστές σε σύγκριση με τα παραδοσιακά μέσα όπως η τηλεόραση και οι εφημερίδες. Έτσι έχει αποκτήσει έκταση τα τελευταία χρόνια και έχει μετατραπεί σε πολυδισεκατομμυριούχα βιομηχανία.

Οι περισσότερες μηχανές αναζήτησης υιοθετούν μια προσέγγιση τριών σταδίων στα χρηματοδοτούμενα προβλήματα αναζήτησης: α)εύρεση των σχετικών διαφημίσεων της αναζήτησης, β)υπολογισμός των κλικ μέσω αναλογίας(CTR) για τις ανακτημένες διαφημίσεις και ιεράρχησή τους και γ)επιλογή του τρόπου που θα εκτεθούν στην σελίδα(π.χ. πόσες διαφημίσεις να δείξουν στο πάνω τμήμα).

Για μια αναζήτηση q (της τάξης 1,...,n),όπου έχουμε ανακτήσει μια σειρά από διαφημίσεις $\{a_1,...,a_n\}$ στην σελίδα αναζήτησης των αποτελεσμάτων, τα αναμενόμενα έσοδα φαίνονται στο τύπο:

$$R=\sum^{n}_{i} P(click|q,a_i)*cost(q',a_i,i) \quad (1)$$

όπου $cost(q',a_i,i)$ είναι το κόστος ενός κλικ για την διαφήμιση a_i για την θέση i για την πληρωμένη φράση q'.

Οι περισσότερες μηχανές αναζήτησης ιεραρχούν τις διαφημίσεις με βάση το αποτέλεσμα του εκτιμώμενου CTR, δηλαδή το $P(click|q,a_i)$, και προσφέρουν χρήματα σε μια προσπάθεια να μεγιστοποιήσουν τα έσοδα. Έτσι το να υπολογίσουμε ακριβώς το CTR για ένα ζευγάρι αναζήτησης -διαφήμισης είναι μια πολύ σημαντική διαδικασία που προσφέρει σημαντικά έσοδα. Μια απλή προσέγγιση είναι να χρησιμοποιήσουμε τα παρατηρημένα CTR για τα ζευγάρια αναζήτησης - διαφήμισης που έχουνε προηγουμένως εμφανιστεί στους χρήστες. Από την άλλη, ο κατάλογος των διαφημίσεων αλλάζει διαρκώς επειδή οι διαφημιστές προσθέτουν, αντικαθιστούν ή διαφοροποιούν τις διαφημίσεις. Έτσι, πολλές αναζητήσεις και διαφημίσεις έχουν λίγες ή καθόλου προηγούμενες "εμφανίσεις" στα αρχεία. Αυτοί οι παράγοντες κάνουν τον υπολογισμό του CTR των σπάνιων και των νέων αναζητήσεων ένα πρόβλημα. Όταν μια σειρά διαφημίσεων έχουν ανακτηθεί και ιεραρχηθεί, η μηχανή αναζήτησης πρέπει να αποφασίσει πόσες διαφημίσεις να δείξει και που να τις τοποθετήσει στην σελίδα. Στην χρηματοδοτούμενη αναζήτηση προτιμούμε να μην δείχνουμε διαφημίσεις όταν το CTR και η συνάφεια είναι χαμηλή.

4.2 Ενσωματώνοντας τα κλικ των χρηστών στην μοντελοποίηση της συνάφειας

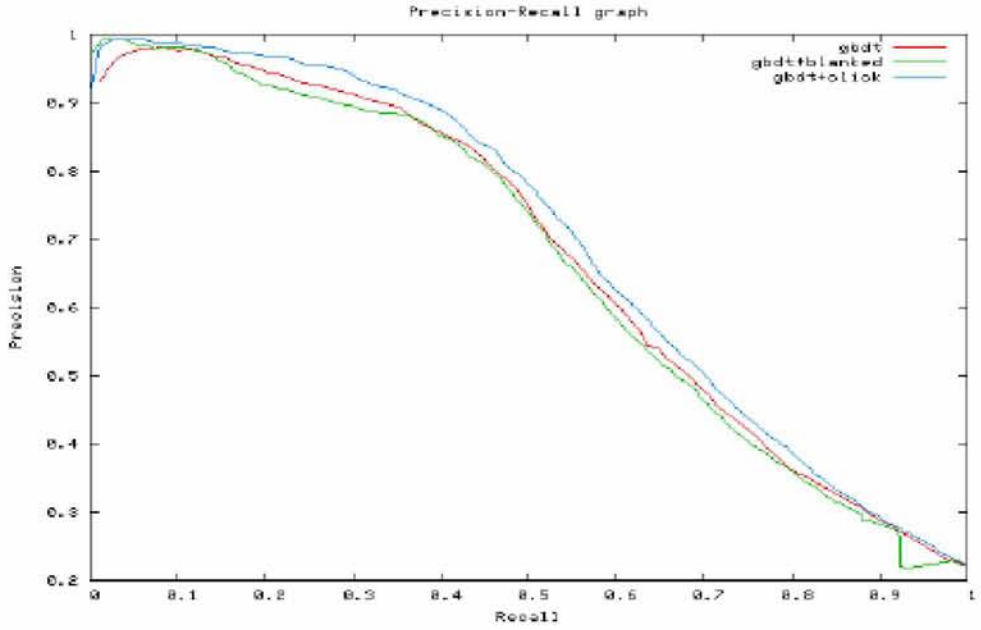
Μαθαίνουμε ένα μοντέλο συνάφειας διαφήμισης που θα μας επιτρέψει να χρησιμοποιήσουμε την προβλεπόμενη συνάφεια για να βελτιώσουμε το σύστημα χρηματοδοτούμενης αναζήτησης. Το μοντέλο συνάφειας είναι ένας δυαδικός ταξινομητής που έχει εκπαιδευτεί να εντοπίζει σχετικές και άσχετες διαφημίσεις, εάν τροφοδοτηθεί με ένα συγκεκριμένο μοντέλο αναζήτησης.

Το βασικό μοντέλο συνάφειας είναι ικανό να προβλέψει την σχετικότητα με μεγάλη ακρίβεια. Βασίζεται σε μια απλή επικάλυψη κειμένου, αλλά αποτυγχάνει να εντοπίσει σχετικές διαφημίσεις, αν δεν υπάρχει συντακτική επικάλυψη. Για παράδειγμα μια διαφήμιση με τίτλο "βρες τα καλύτερα παπούτσια για τζόκινγκ" θα μπορούσε να είναι πολύ σχετική σε μια αναζήτηση για τρέξιμο, αλλά το βασικό μας μοντέλο δεν έχει γνώση ότι το τρέξιμο και το τζόκινγκ έχουν σχέση μεταξύ τους.

4.3 Ιστορικό των κλικ

Το ιστορικό του ρυθμού των κλικ για ένα ζευγάρι αναζήτησης-διαφήμισης μπορεί να παρέχει σαφή ένδειξη συνάφειας και να χρησιμοποιηθεί σαν χαρακτηριστικό στο μοντέλο μας. Όταν δεν υπάρχει ιστορικό κλικ για ένα συγκεκριμένο ζευγάρι, μπορούμε να επιστρέψουμε σε υψηλότερα επίπεδα που ομαδοποιούν το ιστορικό σε ένα γκρουπ διαφημίσεων. Αυτές οι ομαδοποιήσεις επωφελούνται από την συμπεριφορά των παρατηρούμενων κλικ σε παρόμοιες διαφημίσεις από τον ίδιο διαφημιστή και παρέχουν σημαντικά κέρδη στην πιθανότητα να προβλέψουν τα κλικ (περιγράφεται παρακάτω). Αυτά τα χαρακτηριστικά του ιστορικών των κλικ είναι διαθέσιμα μόνο για μια αναλογία διαφημίσεων που έχει επαρκή κίνηση. Οι διαφημίσεις που είναι καινούριες στο σύστημα ή έχουν όρους που δεν χρησιμοποιούνται συχνά δεν θα έχουν αξιόπιστο ιστορικό κλικ. Έτσι είναι σημαντικό να είμαστε σίγουροι ότι προσθέτοντας χαρακτηριστικά κλικ στο μοντέλο συνάφειας, δεν μειώνεται η ακρίβεια του μοντέλου.

Η παρακάτω γραφική παράσταση παρουσιάζει τις καμπύλες ακρίβειας ανάκλησης για τα 3 μοντέλα : το βασικό μοντέλο που έχει μόνο κείμενο, το μοντέλο με κείμενο και χαρακτηριστικά κλικ, και το μοντέλο που έχει εκπαιδευτεί με κείμενο και χαρακτηριστικά κλικ αλλά έχει δοκιμαστεί με τα χαρακτηριστικά κλικ που έχουν σβηστεί.



Καμπύλες ακρίβειας/ανάκλησης όταν λαμβάνεται υπόψιν το ιστορικό των κλικ.-
(Πηγή: Improving ad relevance in Sponsored Search-
Hillard,Schroedl,Manavoglu,Raghavan,Leggeter
atbrox.com/about)

4.4 Η τάση των κλικ κατά τη διαδικασία ερώτημα/απάντηση(διαφήμιση)

Ενώ τα χαρακτηριστικά των κλικ που συζητήθηκαν στο παραπάνω είναι χρήσιμα για διαφημίσεις με επαρκές ιστορικό κλικ, θα μπορούσαμε επίσης να χρησιμοποιήσουμε κάποιες πληροφορίες των κλικ για να μάθουμε σχέσεις που δεν είναι συναφείς με μια συγκεκριμένη διαφήμιση ή ένα συγκεκριμένο διαφημιστή. Παλαιότερη έρευνα(A.Berger,J.Lafferty.Information retrieval at statistical translation .In SIGIR 1999)[1] προτείνει την μοντελοποίηση της αναζήτησης σαν μετάφραση του αρχείου για να ανακτήσουμε πληροφορίες, οπού η συνάφειά του(στην

περίπτωση μιας διαφήμισης) και η ερώτηση αναζήτησης μπορούν να μοντελοποιηθούν με τον κανόνα Bayes σαν :

$$P(D|Q)=p(Q|D)*p(D)/p(Q) \quad (2)$$

Όπου το p(Q) μπορούμε να το αγνοήσουμε επειδή είναι μόνιμο για κάθε συγκεκριμένη αναζήτηση. Ο όρος p(Q|D) μπορεί να θεωρηθεί ένα πρόβλημα στατιστικής μετατροπής και να αποσυντεθεί, χρησιμοποιώντας το μοντέλο IBM 1[8] με τη μορφή:

$$p(Q|D) = \prod_{j=0}^{m} \sum_{i=0}^{n} trans(q_j|d_i) \quad (3)$$

για λέξεις αναζήτησης $q_0,..q_m$ και λέξεις εγγράφων $d_0,...,d_n$ όπου $trans(q_j|d_i)$ είναι μια πιθανότητα συνύπαρξης η οποία έχει συλλεχθεί από μια μάζα παράλληλων ερωτήσεων και αναζητήσεων εγγράφων. Η μεγαλύτερη εκτιμώμενη πιθανότητα των στατιστικών συνεργασίας είναι :

$$trans(q_j|d_i) = \sum_{logs} count(q_j|d_i)/\sum_q \sum_{logs} count(q|d_i)) \quad (4)$$

Η πιθανότητα μετατροπής μετράει τον αριθμό των κλικ που έχει λάβει ένα ζεύγος, διαιρούμενο με τον συνολικό αριθμό των κλικ που έχει λάβει μια λέξη διαφήμισης επί όλες τις λέξεις αναζήτησης. Η λειτουργία καταμέτρησης μπορεί να ενημερωθεί με επαναλήψεις EM(expectation-maximization method που υπολογίζει την μέγιστη πιθανότητα)[1i] όπου ο τύπος $trans(q_j|d_i)$ από την προηγούμενη επανάληψη, "ζυγίζει" τις συνυπάρχουσες μετρήσεις.

Θα μάθουμε δυο μοντέλα μετάφρασης όπου το πρώτο δέχεται τον αριθμό των κλικ, ενώ στη συνέχεια εκπαιδεύουμε το δεύτερο χρησιμοποιώντας στατιστικές που έχουν συλλεχθεί από ζευγάρια αναζήτησης - διαφήμισης. Τα αποτελέσματα "ζυγίζονται" με τα αναμενόμενα κλικ, βασισμένα σε μια διαβαθμισμένη ομαλοποίηση. Για μια διαφήμιση a με τάξη r που έχει ανακτηθεί για μια αναζήτηση q, ορίζουμε το :

$$ec(q,a) = \sum_r imp(q,a,r)P(click|r). \quad (5)$$

Η ποσότητα του ec(q,a) είναι ο αναμενόμενος αριθμός των κλικ, αθροιζόμενος σε όλες τις τάξεις που μια διαφήμιση εμφανίζεται. Η ποσότητα P(click/r) υπολογίζεται παρατηρώντας τον ρυθμό CTR σε μια αναλογία μεγέθους για πολλές μέρες. Ορίζουμε μια αναλογία κλικ:

$$clickLikelihood = p_{click}(Q|D)/pec(Q|D) \quad (6)$$

Αυτή η αναλογία κλικ, παρέχει ένα σκορ το οποίο αφαιρεί την μεροληψία από τα μοντέλα που είναι βασισμένα στα αρχεία.

4.5 Χρηματοδοτούμενες εφαρμογές αναζήτησης

Αυτό το τμήμα περιγράφει σύντομα την προσέγγισή μας στο να αξιολογήσουμε το χρηματοδοτούμενο σύστημα αναζήτησης.

4.5.1 Αξιολόγηση Συστήματος

Η αξιολόγηση ενός live συστήματος αναζήτησης είναι περίπλοκη. Ο μεγάλος αριθμός κίνησης των χρηστών σημαίνει ότι οι ανθρώπινες κρίσεις είναι δύσχρηστες για κάθε αναλογία

δεδομένων. Εναλλακτικά, οι στατιστικές που συλλέγονται από τις αλληλεπιδράσεις των εκατομμυρίων χρηστών είναι διαθέσιμες και μπορούν να παρέχουν σημαντικές ιδέες στην επίδραση μιας πειραματικής προσέγγισης. Έχουμε πρόσβαση σε μια πλατφόρμα που μας επιτρέπει να τρέχουμε το πειραματικό σύστημα για ένα μικρό κομμάτι της κίνησης σε μια μεγάλη εμπορική μηχανή αναζήτησης.

4.5.2 Φιλτράροντας χαμηλής ποιότητας διαφημίσεις

Στόχος του αρχικού σταδίου των χρηματοδοτούμενων συστημάτων αναζήτησης είναι η ανάκτηση ενός υποψήφιου συνόλου σχετικών διαφημίσεων για μια συγκεκριμένη αναζήτηση. Το σύνολο των υποψήφιων διαφημίσεων προέρχεται από διάφορες τεχνολογίες ανάκτησης οι οποίες βασίζονται σε μεθόδους αναδιατύπωσης του ερωτήματος της αναζήτησης καθώς και απευθείας ανάκτηση διαφημίσεων.

Για να βελτιώσουμε την συνάφεια του τελικού υποψήφιου συνόλου, θα εφαρμόσουμε το μοντέλο σχετικότητας σε κάθε ζεύγος ερωτήματος - διαφήμισης(στο υποψήφιο σετ) και θα κόψουμε αυτές που δεν συναντούν το κατώφλι σχετικότητας. Παρακάτω παρουσιάζουμε αποτελέσματα ενός live bucket test που εφαρμόζει online το μοντέλο σχετικότητας σε όλες τις υποψήφιες διαφημίσεις, αφαιρώντας αυτές που δεν συναντούν το κατώφλι σχετικότητας.

Metric	Relative Change
Clicks per candidate set (CTR)	+10.1%
Queries with ads (coverage)	-8.7%
Ads per query (depth)	-11.9%
Clicks per search (Click Yield)	+0.5%

Αποτελέσματα bucket με φίλτρο σχετικότητας
(Πηγή: Improving ad relevance in Sponsored Search-
Hillard,Schroedl,Manavoglu,Raghavan,Leggeter
atbrox.com/about)

Το φίλτρο ελάττωσε σημαντικά τον αριθμό των διαφημίσεων που εμφανίζονται στον χρήστη με μια 8,7% μείωση στα ερωτήματα με διαφημίσεις(κάλυψη) και μείωση 11.9% του μέσου αριθμού των διαφημίσεων ανά ερώτημα αναζήτησης. Ακόμα και με αυτήν την μεγάλη μείωση στον αριθμό των διαφημίσεων, ο μέσος όρος των κλικς ανά αναζήτηση ήταν ουδέτερος, με μια τάση προς το θετικό, το οποίο δείχνει ότι οι περισσότερες από τις διαφημίσεις που έχουν αφαιρεθεί ,δέχτηκαν λίγα κλικ στο βασικό bucket παραγωγής. Μειώνοντας τον αριθμό των διαφημίσεων, παρατηρείται αύξηση 10.1% στο CTR.

Καθώς οι μετρικές σχέσεις των κλικ μας δίνουν μια ένδειξη για το πόσο καλά δουλεύει το φιλτράρισμα του μοντέλου σχετικότητας, ενδιαφερόμαστε να μειώσουμε τις διαφημίσεις χαμηλής σχετικότητας. Ένα online τεστ περιέχει πολλά γεγονότα για να μετρήσουμε τα πάντα λεπτομερώς.

4.5.3 Ιεραρχώντας τις διαφημίσεις με μικρό ιστορικό κλικ.

Διαφημίσεις με μικρό ιστορικό κλικ είναι δύσκολο να ιεραρχηθούν με βάση την πιθανότητα να επιλεχθούν. Σε αυτό το κομμάτι ενσωματώνουμε την προβλεπόμενη σχετικότητα της

διαφήμισης σαν χαρακτηριστικό στην ιεράρχηση, με την πρόθεση να βελτιώσουμε την πρόβλεψη των κλικ(ειδικά όταν μικρό ιστορικό κλικ είναι διαθέσιμο). Οι διαφημίσεις ιεραρχούνται από ένα μοντέλο μηχανής εκμάθησης που προβλέπει την πιθανότητα που έχει μια διαφήμιση ή μια αναζήτηση να επιλεχτεί από τον χρήστη p(click|query,ad). Εισάγουμε ένα μοντέλο μέγιστης εντροπίας για αυτήν την εργασία, το οποίο έχει την ακόλουθη συναρτησιακή μορφή:

$$p(click|query,ad)=1/(1+exp(\sum iwifi) \quad (7)$$

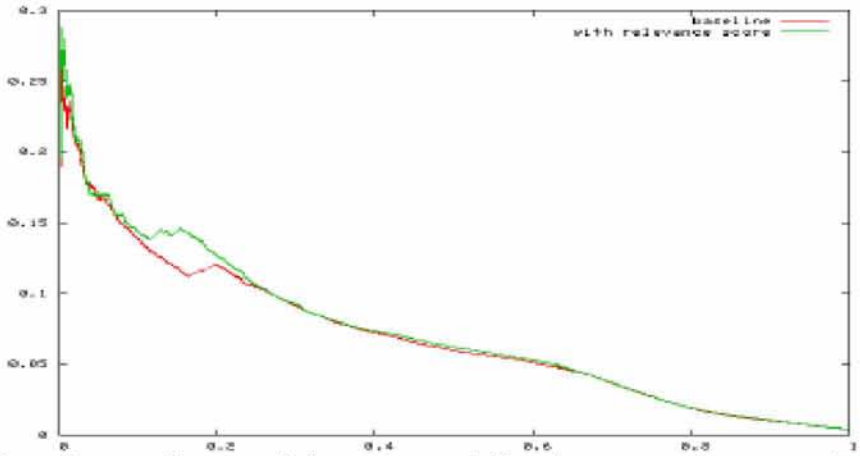
όπου το fi υποδηλώνει ένα χαρακτηριστικό βασιζόμενο στην αναζήτηση ή τη διαφήμιση ή και τα δύο και wi το βάρος σχετιζόμενο με το χαρακτηριστικό. Το μοντέλο περιγράφεται με περισσότερη λεπτομέρεια στη εργασία [5] (Shaparenko,o.Cetin,R.Iyer.Data driven text features for sponsored search click prediction. InAdKWDD Workshop,2009) και εξάγεται από τις καταγραφές των διαφόρων ερωτήσεων. Κάθε γραμμή στην περιοχή καταγραφής των ερωτήσεων συμπεριλαμβάνει ένα ερώτημα και μια διαφήμιση που έχει γίνει κλικ καθώς και άλλες πληροφορίες όπως την ώρα που κλικαρίστηκαν και την θέση στην σελίδα που εμφανίστηκε στον χρήστη. Αυτά τα δεδομένα χρησιμοποιούνται για να εκπαιδεύσουμε έναν δυαδικό ταξινομητή χρησιμοποιώντας το μοντέλο μέγιστης εντροπίας. Τα μοντέλα μέγιστης εντροπίας μπορούν να διαχειριστούν σχετικά καλά, διάσπαρτα και από κοινού συσχετισμένα σύνολα.

Το μοντέλο ιεράρχησης τυπικά αξιολογείται offline με καμπύλες ακρίβειας - ανάκτησης, όπου τα γεγονότα του τεστ είναι χιλιάδες εκατομμύρια κλικ και μη-κλικ. Η απόδοση του μοντέλου είναι πολύ ακριβής όταν είναι διαθέσιμο μεγάλο ιστορικό κλικ. Αυτή η εργασία είναι πιο δύσκολη όταν μικρό ή καθόλου ιστορικό-κλικ είναι διαθέσιμο για μια διαφήμιση. Το βασικό μοντέλο σχετικότητας μπορεί να προβλέψει την συνάφεια, άσχετα από το ιστορικό των κλικ. Έτσι μπορούμε να συμπεριλάβουμε το σκορ σχετικότητας σαν ένα επιπλέον χαρακτηριστικό εισόδου στο μοντέλο. Συγκρίνουμε την απόδοση του βασικού μοντέλου ιεραρχίας των κλικ με ένα μοντέλο ιεράρχησης που ενσωματώνει την προβλεπόμενη σχετικότητα σαν ένα χαρακτηριστικό.

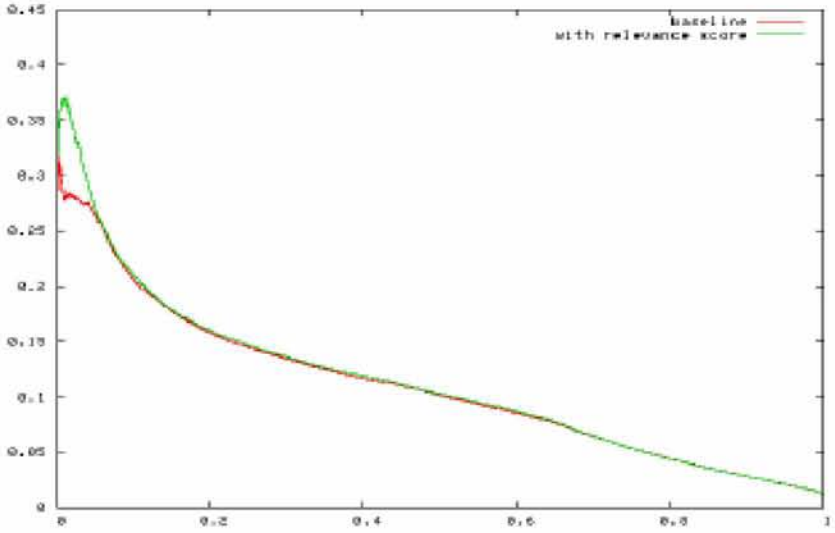
Η πρώτη γραφική παράσταση που ακολουθεί αξιολογεί τα γεγονότα που δεν είχαν λογαριασμό ιστορικού κλικ (1,2% από τα δεδομένα του τεστ). Η δεύτερη γραφική παράσταση που ακολουθεί αξιολογεί τα γεγονότα με καθόλου ιστορικό γκρουπ διαφημίσεων (3,6% από τα δεδομένα του τεστ). Βρίσκουμε ότι το να συμπεριλάβουμε το σκορ σχετικότητας σαν χαρακτηριστικό, παρέχει χρήσιμα οφέλη και στις δυο ρυθμίσεις για τις περιοχές υψηλής ακρίβειας, δείχνοντας ότι οι διαφημίσεις υψηλής ιεραρχίας, ιεραρχούνται με μεγαλύτερη ακρίβεια. Η ακρίβεια για αυτές τις διαφημίσεις βελτιώνεται περισσότερο από 20% για γεγονότα με καθόλου ιστορικό διαφημίσεων. Η περιοχή κάτω από την καμπύλη βελτιώθηκε 3,5% για γεγονότα με καθόλου ιστορικό γκρουπ διαφημίσεων και βελτιώθηκε κατά 5% για γεγονότα με καθόλου ιστορικό λογαριασμού. Τα αποτελέσματα από το υπολειπόμενο test set όπου αρκετά επαρκές ιστορικό-κλικ είναι διαθέσιμο δεν έχουν αλλάξει από το βασικό μοντέλο ιεράρχησης των κλικ.

Ο πίνακας που ακολουθεί συγκρίνει το CTR των δυο μοντέλων σε ερωτήματα με διαφορετικά μεγέθη ιστορικού κλικ. Παρουσιάζουμε τα αποτελέσματα για τις δυο κορυφαίες θέσεις επειδή τα μεγέθη των δειγμάτων για τις χαμηλότερες θέσεις δεν ήταν αρκετά μεγάλα έτσι ώστε να παρουσιάζουν σημαντικές αλλαγές.

Το μοντέλο που περιλαμβάνει την προβλεπόμενη σχετικότητα σαν χαρακτηριστικό έχει αυξήσει και την τάξη 1 και την τάξη 2 CTR σημαντικά κατά 12,7% και 16,9% αντίστοιχα για το κομμάτι του χαμηλού ιστορικού. Δεν παρατηρήσαμε καμία αλλαγή για τις αναζητήσεις που είχαν αρκετές πληροφορίες ιστορικού. Σημειώστε ότι αυτά τα ευρήματα είχαν προβλεφθεί από την offline ανάλυσή μας που παρουσιάσαμε παραπάνω. Όλως παραδόξως δεν είδαμε τις βελτιώσεις που είχαν προβλεφθεί για το κομμάτι της αναζήτησης με καθόλου ιστορικό. Αυτό ίσως να οφείλεται κατά ένα μέρος στο μικρό μέγεθος των δειγμάτων που συλλέξαμε για αυτό το σύνολο ερωτήσεων και κατά ένα άλλο μέρος στην ίδια τη φύση των ερωτήσεων.



Καμπύλη ακρίβειας-ανάκλησης με καθόλου λογαριασμό ιστορικού κλικ
(Πηγή: Improving ad relevance in Sponsored Search-
Hillard,Schroedl,Manavoglu,Raghavan,Leggeter
atbrox.com/about)



Καμπύλη ακρίβειας-ανάκλησης με καθόλου ιστορικό γκρουπ διαφημίσεων.
(Πηγή: Improving ad relevance in Sponsored Search-
Hillard,Schroedl,Manavoglu,Raghavan,Leggeter
atbrox.com/about)

	Click history levels		
	None	Low History	High History
Rank1 CTR	+0.1%	+12.7% ($p < 0.05$)	-0.5%
Rank2 CTR	+2.8%	+16.9% ($p < 0.1$)	+1.3%

ΠΙΝΑΚΑΣ –Αποτελέσματα bucket
(Πηγή: Improving ad relevance in Sponsored Search-
Hillard,Schroedl,Manavoglu,Raghavan,Leggeter
atbrox.com/about)

4.6 Μείωση της επίδρασης(στον χρήστη) των "βόρειων" διαφημίσεων

Αφού έχει δοθεί ένα ιεραρχημένο σύνολο υποψήφιων διαφημίσεων, το τελικό στάδιο της χρηματοδοτούμενης αναζήτησης πρέπει να αποφασίσει πόσες διαφημίσεις να τοποθετήσει στο πάνω μέρος. Η τοποθέτηση των διαφημίσεων πάνω από τα αποτελέσματα αναζήτησης (παρά στο πλάι δεξιά) δημιουργεί ένα άμεσο ανταγωνισμό μεταξύ των διαφημίσεων και των αποτελεσμάτων της αναζήτησης. Οι διαφημίσεις μπορούν να αποπροσανατολίσουν τους χρήστες και ιδίως να τους αποτρέψουν να βρούνε σελίδες που περιέχουν την πληροφορία που ζητήσανε. Η μηχανή αναζήτησης μπορεί σκόπιμα να προκαλέσει υποβάθμιση της εμπειρίας του χρήστη, με σκοπό να έχει τα αναμενόμενα έσοδα. Οι διαφημίσεις που δεν εμφανίζονται στο πάνω μέρος, μπορούν να εμφανιστούν δεξιά ή κάτω. Παρόλα αυτά, το μέγεθος της επίδρασης στην εμπειρία του χρήστη και των εσόδων, προέρχεται από τις πάνω διαφημίσεις εξαιτίας της εξέχουσας θέσης τους στην σελίδα. Ένας τρόπος να μετρήσουμε την ποιότητα ανάκτησης της αναζήτησης είναι το DCG(Discounted Cumulative Gain)[3](K.Jarvelin ,J.kekalainen.Cumulated gain-baised evaluation for ir techniques):

$$DCGn=\sum_{i=1}^{n} w_i rel_i$$

Όταν οι διαφημίσεις που τοποθετούνται πάνω από τα αποτελέσματα αναζήτησης υποβαθμίζουν την ποιότητα της ανάκτησης, τότε η υποβάθμιση μπορεί να μετρηθεί ως NAI.Η ποσοστιαία μείωση στο DCG εισάγεται από τις προβαλλόμενες διαφημίσεις ως:

$$\mathbf{NAI=(DCG_{NO\ ADS}-DCG_{WITH\ ADS})/DCG_{NO\ ADS}}$$

Μπορούμε να επιχειρήσουμε να μειώσουμε το NAI στο σύστημα αναζήτησής μας, υπολογίζοντας το DCG(μέτρο της αποτελεσματικότητας του αλγορίθμου μηχανής αναζήτησης που χρησιμοποιείται συνήθως για εξόρυξη γνώσης.) πριν και μετά τις δυνατές θέσεις της διαφήμισης(στο επάνω μέρος) και διαλέγοντας να τοποθετήσουμε τις διαφημίσεις στο συγκεκριμένο επάνω μέρος όπου θα επιβαρυνθούμε με την μικρότερη NAI τιμωρία.

Η χρηματοδοτούμενη έρευνα προσφέρει έναν προκαθορισμένο αριθμό διαφημίσεων ανά ερώτηση (βασιζόμενοι στα αναμενόμενα κέρδη) και μετά διαλέγει πότε να τοποθετήσει διαφήμιση στο πάνω μέρος για μια συγκεκριμένη αναζήτηση βελτιώνοντας κάποια κριτήρια. Το βασικό μας σύστημα βελτιώνει την τοποθέτηση διαφήμισης στο πάνω μέρος βασιζόμενο σε ένα συνδυασμό από τα μέγιστα έσοδα (η πρόβλεψη της διαφήμισης για το πόσες φορές θα κλικαριστεί μια διαφήμιση) και το κόστος χρήσης (μια τιμωρία για χαμηλής ποιότητας διαφημίσεις). Οι διαφημίσεις ταξινομούνται από την συνάρτηση βελτιστοποίησης και έπειτα τοποθετούνται στο πάνω μέρος αν το σκορ τους είναι πάνω από το κατώφλι που παράγει ένα κατά μέσο όρο ποσοστό

των διαφημίσεων στο πάνω μέρος.

4.7 Συμπεράσματα

Έχουμε παρουσιάσει ένα βασικό μοντέλο σχετικότητας που προβλέπει με ακρίβεια τη συνάφεια για ζεύγη ερωτήματος-διαφήμισης και επιπρόσθετα έχουμε βελτιώσει αυτό το μοντέλο ενσωματώνοντας έμμεση τροφοδοσία από τα διάσπαρτα κλικ των χρηστών. Βρήκαμε ότι το παρατηρημένο ιστορικό των κλικ μας βοηθάει στο να προβλέψουμε τη σχετικότητα όταν επαρκείς διαφημίσεις είναι διαθέσιμες. Όταν λίγες ή καθόλου παρατηρήσεις είναι διαθέσιμες, παρουσιάσαμε μια μέθοδο για την εκμάθηση ενός μοντέλου μετάφρασης, από καταγραφές οι οποίες μπορούν να γενικευτούν σε μη-εμφανείς διαφημίσεις. Και οι δυο προσεγγίσεις παρέχουν σημαντικές βελτιώσεις στο να προβλέψουμε την σχετικότητα .

Μετά εφαρμόσαμε το μοντέλο σχετικότητας σε τρία μεγάλα συστατικά ενός συστήματος χρηματοδοτούμενης αναζήτησης : ανάκτηση διαφημίσεων, ιεράρχηση διαφημίσεων, τοποθέτηση στην σελίδα.

Βιβλιογραφία

[1] A. Berger and J.Lafferty, "Information retrieval as statistical translation," In SIGIR 1999.
[2] P. F. Brown, V. J. Pietra, S. A. D. Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," Computational Linguistics, 1993.
[3] K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of ir techniques," ACM Trans. Inf. Syst., 2002.
[4] M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks: estimating the click-through rate for new ads," In WWW, 2007.
[5] B. Shaparenko, O. Cetin, and R. Iyer, "Data driven text features for sponsored search click prediction," In AdKDD Workshop, 2009.

D. Hillard, S. Schroedl, E. Manavoglu, H. Raghavan, C. Leggeter, "Improving ad relevance in Sponsored Search," New York 2010, www.atbrox.com/about

5.Εξαγωγή προφίλ χρήστη από μεγάλο όγκο δεδομένων

Ένα προφίλ χρησιμοποιείται για να κατηγοριοποιήσει ένα συγκεκριμένο χρήστη σε προκαθορισμένα τμήματα ή να αποτυπώσει την online συμπεριφορά του χρησιμοποιώντας τα ενδιαφέροντα και τις προτιμήσεις του. Μπορεί να οριστεί από τον ίδιο τον χρήστη κατά την διάρκεια της εγγραφής του σε κάποια υπηρεσία. Συχνά ορίζεται σαν την επεξεργασία δεδομένων που συνδέονται με αυτόν. Οι πηγές δεδομένων για το προφίλ περιλαμβάνουν τις περιηγήσεις στο διαδίκτυο, τα περιεχόμενα του χρήστη(π.χ. κάποιο blog του), τον ρυθμό των κλικ που εξάγονται από τις εγγραφές των αναζητήσεων ή άλλα προφίλ χρηστών χρησιμοποιώντας τεχνικές φιλτραρίσματος. Τα προφίλ εξελίσσονται στον χρόνο λόγω των πιθανών αλλαγών στα ενδιαφέροντα και τα γούστα των χρηστών. Ο σκοπός μας είναι να αναπτύξουμε μια λύση με ένα κλιμακωτό προφίλ, το οποίο αιχμαλωτίζει αυτήν την εξέλιξη και διατηρεί τα ποιοτικά χαρακτηριστικά του εκάστοτε χρήστη.

Τα προφίλ χρησιμοποιούνται συνήθως για τον σκοπό της εξατομίκευσης. Για παράδειγμα μπορούν να χρησιμοποιηθούν για συγκεκριμένη αναζήτηση του χρήστη, ξανακατατάσσοντας τα αποτελέσματα σύμφωνα με το ερώτημα του χρήστη. Άλλο ένα παράδειγμα είναι τα online μαγαζιά όπως το Amazon και Shopping.com τα οποία χρησιμοποιούν τα προφίλ των χρηστών για συγκεκριμένες προτάσεις προϊόντων σε πιθανούς online αγοραστές. Πρόσφατα στο περιεχόμενο των online εκστρατειών διαφήμισης ,το προφίλ χρήστη χρησιμοποιείται από μια ποικιλία λύσεων στοχευμένης συμπεριφοράς για την βελτίωση του CTR των διαφημίσεων.

Θα αντιμετωπίσουμε το πρόβλημα της εξόρυξης και της διατήρησης ενός πολύ μεγάλου αριθμού προφίλ χρηστών από πολύ μεγάλης κλίμακας δεδομένων με υψηλής ποιότητας εγγυήσεις. Στην συνέχεια περιγράφουμε μια κλιμακωτή εφαρμογή από το προτεινόμενο framework στο Apache Hadoop και συζητάμε τις προκλήσεις και τους σκοπούς του. Βασικό μας μέλημα είναι να προτείνουμε ένα κλιμακωτό framework για την επιλογή χαρακτηριστικών γνωρισμάτων όπου το προφίλ του χρήστη εκπροσωπείται από τα περιεχόμενα του κειμένου τα οποία παράγονται από διαφορετικούς χρήστες και ο σκοπός μας είναι να "ζυγίσουμε " το προφίλ σύμφωνα με την ικανότητα να παρουσιάσουμε τα ενδιαφέροντά του. Για αυτόν τον σκοπό προτείνουμε μια αποδοτική και κλιμακωτή επιλογή χαρακτηριστικών στην μέθοδό μας ,βασισμένη στην Kullback-Leibler (KL) απόκλιση, προσαρμοσμένη στης ανάγκες των εργασιών του χρήστη. Στην συνέχεια παρουσιάζουμε πώς το συγκεκριμένο πλαίσιο μπορεί να εφαρμοστεί στο Apache Hadoop MapReduce .Χρησιμοποιώντας τα δεδομένα του πραγματικού κόσμου, αξιολογούμε την κλιμάκωση του πλαισίου. Εν κατακλείδι, διαπιστώνουμε δύο κύριες συνεισφορές: καταρχάς περιγράφουμε ένα αποδοτικό πλαίσιο προφίλ χρήστη με υψηλής ποιότητας εγγυήσεις και κατά δεύτερον μια κλιμακωτή εφαρμογή του προτεινόμενου πλαισίου στο Apache Hadoop.

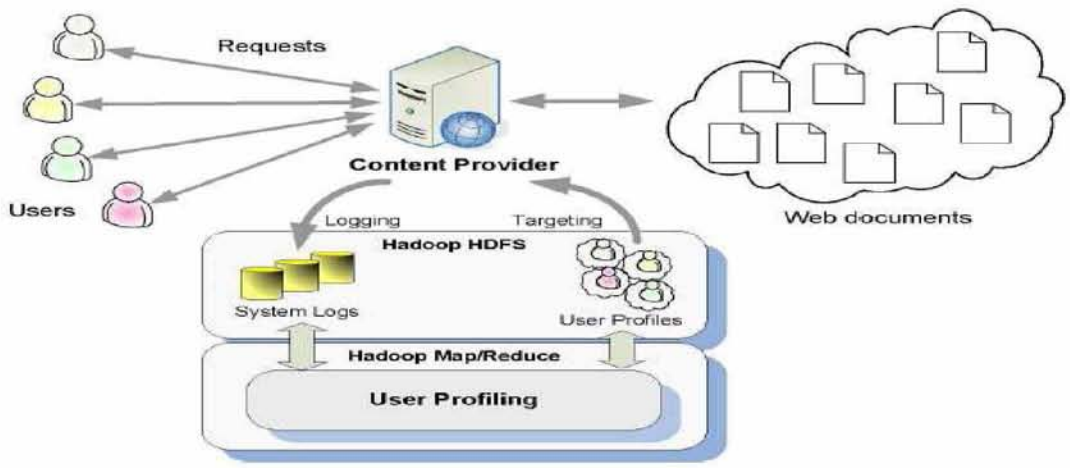
5.1 Πλαίσιο σκιαγράφησης του χρήστη

Θα περιγράψουμε ένα πλαίσιο μεγάλης κλίμακας το οποίο περιέχει λεπτομέρειες για τον χρήστη. Ξεκινάμε περιγράφοντας το γενικότερο πλαίσιο ακολουθούμενο από τις λεπτομέρειες του μοντέλου. Μετά δείχνουμε πώς τα προφίλ των χρηστών εξάγονται και κλείνουμε την ενότητα με μια περιγραφή του πλαισίου εφαρμογής στο Hadoop.

5.2 Ρύθμιση(setting)

Στην παρακάτω εικόνα θεωρούμε ένα σύνολο διαχείρισης γενικού περιεχομένου με πολλούς χρήστες, οι οποίοι υποβάλουν διάφορες αιτήσεις για έγγραφα του διαδικτύου σε κάποιον πάροχο. Ο εξυπηρετητής του παρόχου καταγράφει κάθε αίτηση στο σύστημα το οποίο με τη σειρά του καταχωρεί όλες τις αλληλεπιδράσεις των χρηστών με τα έγγραφα του ενδιαφέροντός τους. Κάθε εγγραφή είναι μια πλειάδα (u, d, context) και περιλαμβάνει την σχέση χρήστη-εγγράφου όπου το u αντιπροσωπεύει τον χρήστη(user), το d αντιπροσωπεύει το έγγραφο(document) που σχετίζεται με αυτόν, και context είναι επιπρόσθετα δεδομένα που έχουν εξαχθεί από το γενικότερο πλαίσιο της σχέση χρήστη-εγγράφου(π.χ. χρόνος, γεωγραφική τοποθεσία).Η ενότητα της σκιαγράφησης του χρήστη χρησιμοποιεί τα καταγεγραμμένα δεδομένα του συστήματος κατά την διάρκεια προγραμματισμένων χρονικών περιόδων(π.χ. μια φορά την ημέρα) και είναι υπεύθυνο να διατηρεί τα προφίλ των χρηστών. Οι εγγραφές του συστήματος και τα προφίλ των χρηστών αποθηκεύονται στο σύστημα αρχείων (filesystem) του Hadoop (HDFS). Με το που αναβαθμίζονται τα προφίλ των χρηστών σε μια χρονική περίοδο j όλες οι επιμέρους εγγραφές διαγράφονται. Έτσι, σε κάθε χρονική περίοδο j για να αναβαθμίσουν τα προφίλ των χρηστών χρειάζονται οι εγγραφές ανάμεσα στην προηγούμενη χρονική περίοδο j-1 και στην τρέχουσα.

Τα προφίλ χρηστών μπορούν να χρησιμοποιηθούν από τον πάροχο περιεχομένων για στοχευμένες υπηρεσίες(π.χ. Διαφημίσεις, προτάσεις) ή για να παρέχουν εξατομικευμένες υπηρεσίες(π.χ. Εξατομίκευση αποτελεσμάτων αναζήτησης ενός χρήστη). Στην συνέχεια εστιάζουμε στις πλευρές του μοντέλου προφίλ χρήστη και της μεγάλης κλίμακας εφαρμογές στο Hadoop.



Σύνολο διαχείρισης γενικού περιεχομένου
Πηγή:M.Shmueli-Scheuer,H.Roitman,D.Carmel,Y.Mass,D.Konopnicki,Extracting User Profiles from Large Scale
Atbrox.com/about

5.3 Μοντέλο προφίλ χρήστη.

5.3.1 Βασικοί συμβολισμοί

Δεδομένης κάποιας χρονικής περιόδου j δηλώνουμε με D_j το "στιγμιότυπο κοινότητας"(community snapshot) καθορισμένο ως το σύνολο όλων των εγγράφων d που έχουν αναζητηθεί, τα οποία έχουν καταγραφεί στο σύστημα κατά την διάρκεια αυτής της περιόδου. Δεδομένου ενός χρήστη u δηλώνουμε επίσης με $D_j(u)$ το "στιγμιότυπο χρήστη", ορισμένο ως το υποσύνολο των εγγράφων $d \in D_j$ που είχαν αναζητηθεί από τον χρήστη u κατά την διάρκεια της χρονικής περιόδου j.

Ένα προφίλ χρήστη αντιπροσωπεύεται από το περιεχόμενο των εγγράφων κειμένου που σχετίζονται με αυτόν. Σε αυτήν την εργασία προσαρμόζουμε το μοντέλο Bag Of Words(BOW)(L. Chen and K. Sycara. Webmate: a personal agent for browsing and searching. In AGENTS '98, K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users) για να αντιπροσωπεύσουμε τα προφίλ χρηστών. Δεδομένου ενός λεξιλογίου ορών $V=\{t_1,t_2,....,t_m\}$,το προφίλ κάθε χρήστη u σε μια δεδομένη χρονική περίοδο j (που υποδηλώνεται από $p_j(u)$),ορίζεται ως διάνυσμα βάρους:

$$p_j(u) = (w_j^u(t_1), w_j^u(t_2), \ldots, w_j^u(t_m))$$

όπου κάθε διάνυσμα $w_j^u(t)$ αντιστοιχεί στο προφίλ ενός μοναδικού όρου $t \in V$ σε μια χρονική περίοδο j ,το οποίο μπορεί να περιέχεται στο κείμενο των συσχετιζόμενων εγγράφων $D_j(u)$.Αν το περιεχόμενο του προφίλ χρήστη δεν περιέχει τον όρο t ,τότε $w_j^u(t)=0$.Κάθε όρος μπορεί να είναι είτε λέξη είτε φράση.

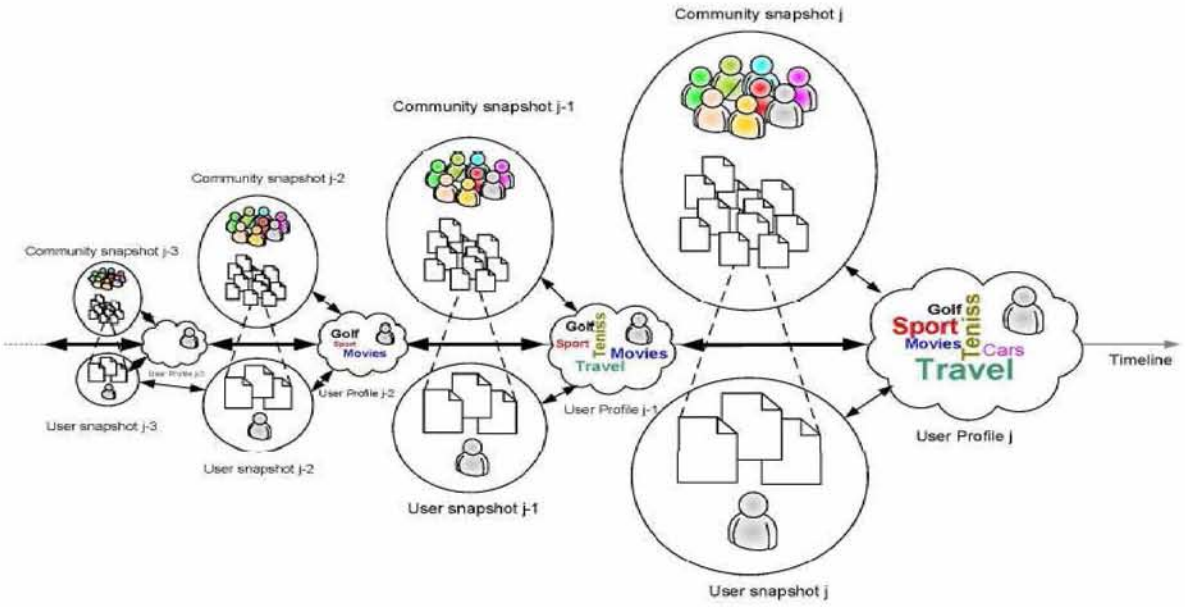
5.3.2 Διατήρηση προφίλ χρήστη

Για να απεικονίσουμε ένα στιγμιότυπο του προφίλ χρήστη σε μια δεδομένη χρονική στιγμή, χρησιμοποιούμε τον ακόλουθο αναδρομικό κανόνα αναβάθμισης:

$$\tilde{p}_j(u) \leftarrow \alpha_j \cdot p_{j-1}(u) + (1 - \alpha_j) \cdot p_j(u) \qquad (1)$$

όπου a_j είναι μια παράμετρος που ελέγχει την σχετική προτίμηση του συστήματος ανάμεσα στο πιο πρόσφατο στιγμιότυπο προφίλ χρήστη $p_j(u)$ και το ιστορικό προφίλ $p_{j-1}(u)$,όπου $p_o(u)$ εισάγεται με μηδενικά βάρη. Επιπλέον, χρησιμοποιούμε εκθετική εξομάλυνση και θέτουμε $a_j=a_o*\exp^{-j/C}$.Όσο μεγαλύτερο είναι το a_j,τόσο πιο πολύ βασιζόμαστε στο προφίλ του χρήστη που έχουμε μάθει από το ιστορικό(μοντέλο εκμετάλλευσης) και τόσο λιγότερο βασιζόμαστε στο πρόσφατο προφίλ(μοντέλο εξερεύνησης).Η παράμετρος C ελέγχει τον ρυθμό εκμάθησης όπου μεγαλύτερες τιμές υπονοούν πιο αργή διαδικασία εκμάθησης. Μαθαίνουμε και τις δυο παραμέτρους από δείγματα και ορίζουμε $a_o=0.1$ και $C=100$.

Το επόμενο σχήμα μας δείχνει ένα παράδειγμα ενός εξελισσόμενου προφίλ χρήστη $p_j(u)$ κάποιου χρήστη u κατά την διάρκεια διαφορετικών χρονικών περιόδων j.Σε κάθε χρονική περίοδο, το προφίλ χρήστη παρουσιάζεται στο σχήμα 2 με ένα συννεφάκι ,όπου τα πιο πρόσφατα στιγμιότυπα παρουσιάζονται με μεγαλύτερα συννεφάκια. Το στιγμιότυπο κοινωνίας D_j και το στιγμιότυπο του χρήστη $D_j(u)$ σε κάθε χρονική περίοδο j φαίνονται στο σχήμα 2 με μια έλλειψη (επάνω) και ένα κύκλο(κάτω).



Παράδειγμα εξέλιξης προφίλ χρήστη στο χρόνο
Πηγή:M.Shmueli-Scheuer,H.Roitman,D.Carmel,Y.Mass,D.Konopnicki,Extracting User Profiles from Large Scale
Atbrox.com/about

5.4 Σταθμίζοντας τους όρους του προφίλ χρήστη

Το "στάθμισμα" των όρων του προφίλ χρήστη σχετίζεται άμεσα με την επιλογή χαρακτηριστικών(feature selection) που είναι η διαδικασία επιλογής ενός υποσυνόλου όρων για αναπαράσταση κειμένου και συχνά εφαρμόζεται στις μεθόδους κατηγοριοποίησης κειμένου και μεθόδους clustering [5]. Οι προσεγγίσεις για την επιλογή χαρακτηριστικών αξιολογούν τους όρους σύμφωνα με την ικανότητά τους να ξεχωρίζουν το κείμενο που μας δίνεται από το ολόκληρο. Για την αποτύπωση εργασιών του χρήστη δοθείσης μιας περιόδου j,χρησιμοποιούμε τα πιο πρόσφατα χαρακτηριστικά του χρήστη u ζυγίζοντας τους όρους στο $p_j(u)$.

Στην συνέχεια περιγράφουμε μια μέθοδο επιλογής χαρακτηριστικών η οποία είναι βασισμένη σε λίγες, απλές αρχές όπως website findability[2] και cluster labeling[1]. Συζητάμε το σύνολο των βασικών χαρακτηριστικών που θα χρησιμοποιήσουμε και στην συνέχεια το πως οι όροι του προφίλ χρήστη σταθμίζονται χρησιμοποιώντας την προτεινόμενη μέθοδο επιλογής χαρακτηριστικών που βασίζεται στην μέθοδο KL[4].

5.4.1 Χαρακτηριστικά όρων

Η μέθοδος προφίλ χρήστη, είναι βασισμένη σε αρκετά χαρακτηριστικά χαμηλού επιπέδου IR τα οποία εξάγονται από τα στιγμιότυπα Dj και Dj(u).Δοθέντος ενός όρου $t \in V$ και ενός αρχείου $d \in Dj(u)$, το πρώτο χαρακτηριστικό είναι ο όρος συχνότητας t στο αρχείο d ,που επισημαίνεται στο

tf(t,d).Το δεύτερο χαρακτηριστικό είναι ο όρος συχνότητας εγγράφου t, ορισμένος ως ο αριθμός των εγγράφων σε μια δεδομένη συλλογή εγγράφων τα οποία περιέχουν τον όρο t. Για την χρονική περίοδο j, υποδηλώνουμε την συχνότητα του αντικειμένου t με $df(t,D_j)$ και $df(t,D_j(u))$ στο j-στό στιγμιότυπο.

Διαπιστώνουμε ότι ο όρος t που έχει υψηλή συχνότητα εμφάνισης στα αντικείμενα $d \in D_j(u)$, θα πρέπει να εμφανίζεται με την ίδια μεγάλη πιθανότητα όπως στο στιγμιότυπο $D_j(u)$ και με την ίδια μικρή πιθανότητα όπως στο στιγμιότυπο D_j. Με αυτήν την διαπίστωση, και δοθέντος του όρου $t \in V$ και των στιγμιότυπων $D_j(u)$ και D_j, το βάρος δίνεται από τον παρακάτω τύπο:

$$w_j^u(t)=tf(t,D_j(u))udf(t,D_j(u))idf(t,D_j) \quad (2)$$

όπου: $tf(t,D_j(u))=\Sigma_{d \in Dj(u)}tf(t,d)/D_j(u)$ είναι η μέση συχνότητα του όρου $D_j(u)$
$udf(t,D_j(u))=df(t,D_j(u))/|D_j(u)|$ είναι η μέγιστη πιθανότητα εύρεσης του όρου t στο $D_j(u)$ και
$idf(t,D_j)=\log(1+ |D_j|/df(t,D_j))$ είναι η αντίστροφη τιμή της συχνότητας εμφάνισης του όρου t στο έγγραφο, η οποία καθορίζεται από την συχνότητα του j-στου στιγμιότυπου D_j.

5.4.2 Μέθοδος KL.

Θυμηθείτε ότι δοθείσης μια χρονικής περιόδου j προσπαθούμε να συγκεντρώσουμε όλα τα πρόσφατα ενδιαφέροντα του χρήστη u σταθμίζοντας τους όρους του $p_j(u)$ σύμφωνα με την συνεισφορά στο τωρινό στιγμιότυπο $D_j(u)$ από το στιγμιότυπο D_j. Αυτή η διαχωριστικότητα μπορεί να μετρηθεί χρησιμοποιώντας Kullback - Leibler (KL) διαφορά [4] ανάμεσα στην κατανομή στιγμιότυπου $D_j(u)$ και D_j. Για τις δύο κατανομές $P_1(t)$ και $P_2(t)$ για τους όρους της συλλογής $t \in V$,ο KL διαχωρισμός ορίζεται ως:

$$\mathfrak{D}_{KL}(P_1||P_2) = \sum_{t \in V} P_1(t) \log \frac{P_1(t)}{P_2(t)} \quad (3)$$

Για ένα δεδομένο όρο $t \in V$, έστω $P(t|D_j(u))$ και $P(t|D_j)$ δηλώνουν τις οριακές κατανομές των όρων για τα σύνολα των εγγράφων $D_j(u)$ και D_j αντίστοιχα. Τώρα μπορούμε να περιγράψουμε πως δημιουργήθηκαν οι δυο κατανομές. Δεδομένου ενός στιγμιότυπου Dj, η οριακή κατανομή του όρου t υπολογίζεται ως:

$$P(t|D_j) = \overline{tf}(t, D_j)cdf(t, D_j)N_j \quad (4)$$

όπου $cdf(t,D_j)=df(t,D_j)/|D_j|$ είναι η μέγιστη εκτιμώμενη πιθανοφάνεια της πιθανότητας να βρούμε τον όρο t στο Dj και $Nj=\Sigma_{t \in V}P(t|Dj)^{-1}$ είναι ο παράγοντας κανονικοποίησης της πιθανότητας.

Για το στιγμιότυπο Dj(u) χρησιμοποιούμε την εξίσωση (2), και μετράμε την οριακή κατανομή βασιζόμενοι στον όρο t και στο βάρος των όρων το οποίο περιγράφεται στον ακόλουθο τύπο:

$$P(t|D_j(u)) = (1-\lambda)\left(\frac{w_j^u(t)}{\sum_{t \in V} w_i^u(t)}\right) + \lambda P(t|D_j) \quad (5)$$

όπου λ είναι η παράμετρος που χρησιμοποιείται για να εξομαλύνουμε τον όρο $P(t,D_j(u))$ με τον $P(t|D_j)$. Αυτό αποτελεί μια κοινή τεχνική στα μοντέλα γλωσσών[5]. Σε αυτήν την εργασία, θεωρούμε το $\lambda=0.001$.

Τέλος, υπολογίζουμε τον όρο t του προφίλ κάθε χρήστη από την οριακή κατανομή του αυξάνοντας τον όρο KL ανάμεσα στην κατανομή του στιγμιότυπου χρήστη και κοινότητας. Αυτό

υπολογίζεται από τον ακόλουθο τύπο:

$$\tilde{w}_j^u(t) = P(t|D_j(u)) \log \frac{P(t|D_j(u))}{P(t|D_r)}$$

5.5 Εφαρμογή μεγάλης κλίμακας

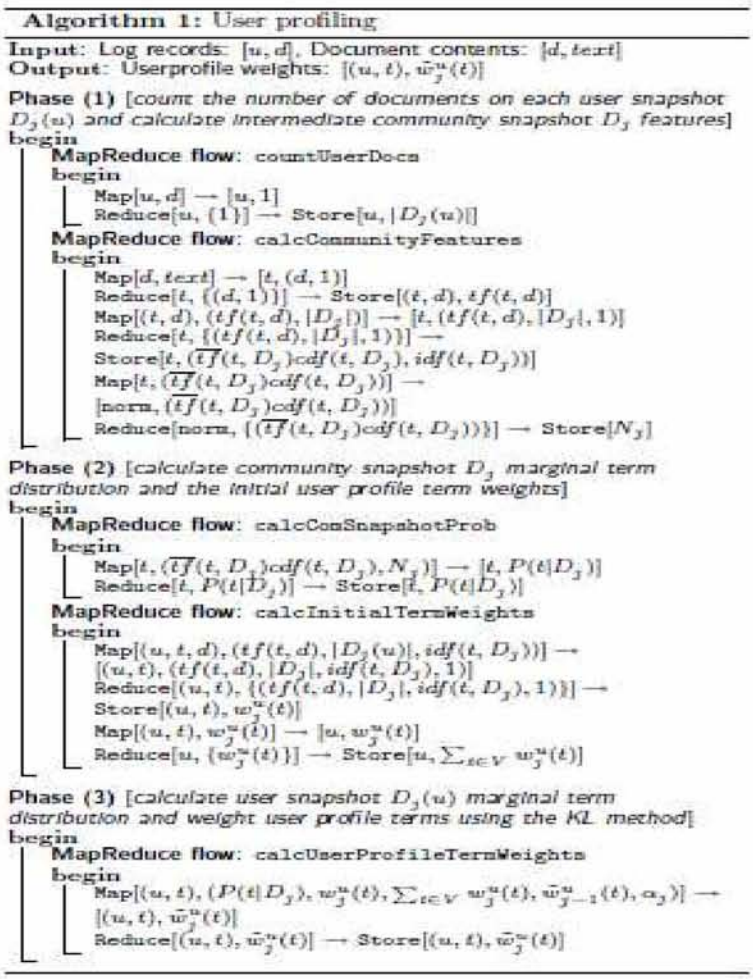
Περιγράφουμε τώρα μια αποδοτική υλοποίηση μεγάλης κλίμακας στο Hadoop MapReduce η οποία φαίνεται στον ψευδο-κώδικα. Έχουμε δύο υπολογιστικά σημεία συμφόρησης εκ των οποίων το πρώτο οφείλεται στον υπολογισμό της πιθανότητας κανονικοποίησης του παράγοντα N_j, που χρησιμοποιήθηκε για την κανονικοποίηση της πιθανότητας του στιγμιότυπου στην εξίσωση (4) και το δεύτερο σημείο συμφόρησης που προήλθε από τον υπολογισμό των βαρών του κανονικοποιημένου παράγοντα $\Sigma_{t\varepsilon V} w_j^u(t)$ στην εξίσωση (5).

Ο αλγόριθμος χωρίζεται σε τρεις φάσεις. Κάθε φάση ξεκινάει ενσωματώνοντας τα ενδιάμεσα αποτελέσματα της προηγούμενης φάσης από το HDFS, ακολουθούμενα από την παράλληλη εκτέλεση της λογικής φάσης, χρησιμοποιώντας ροές Hadoop MapReduce και τελειώνει αποθηκεύοντας τα ενδιάμεσα αποτελέσματα στο HDFS για χρήση από τις φάσεις που ακολουθούν. Στη συνέχεια περιγράφουμε τη ροή της κάθε φάσης με περισσότερες λεπτομέρειες. Οι συμβολισμοί Map[(key1), (val1)] → [(key2), (val2)] και Reduce[(key2), {(val2)}] → [(key3), (val3)] στον Αλγόριθμο 1 υποδηλώνουν μια υποροή MapReduce ενώ ο συμβολισμός Store[(key), (val)] υποδηλώνει την αποθήκευση ενδιάμεσων αποτελεσμάτων στην HDFS.

Η πρώτη φάση εκτελεί δύο παράλληλες ροές MapReduce. Η πρώτη ροή countUserDocs, παίρνει σαν είσοδο ένα ζεύγος(χρήστη - εγγράφου) και μετράει για κάθε χρήστη τον αριθμό των αρχείων στο συγκεκριμένο στιγμιότυπο. Η δεύτερη ροή, η calcCommunityFeatures ξεκινά με μια MapReduce ροή, που υπολογίζει την συχνότητα κάθε όρου σε κάθε αρχείο tf(t,d), του οποίου η έξοδος αποθηκεύεται στο HDFS για χρήση στην επόμενη φάση και διοχετεύεται σε μια άλλη ροή MapReduce η οποία υπολογίζει τα χαρακτηριστικά του στιγμιοτύπου για κάθε όρο. Η έξοδος όλων αποθηκεύεται στο HDFS για χρήση σε επόμενη φάση και διοχετεύεται στο MapReduce, το οποίο υπολογίζει την τιμή του κανονικοποιημένου παράγοντα Nj και την αποθηκεύει στο HDFS για την επόμενη φάση.

Η δεύτερη φάση εκτελεί επίσης δυο παράλληλες ροές MapReduce. Η πρώτη ροή calcComSnapshotProb, παίρνει τα ενδιάμεσα χαρακτηριστικά του στιγμιοτύπου της κοινότητας τα οποία υπολογίστηκαν στην προηγούμενη φάση και υπολογίζει για κάθε όρο $t \in V$ την πιθανότητα P(t|Dj), σύμφωνα με την εξίσωση (4).Οι τιμές της πιθανότητας αποθηκεύονται στο HDFS για μεταγενέστερη χρήση στην επόμενη φάση. Η δεύτερη ροή calcInitialTermWeights, αρχίζει με μια ροή MapReduce η οποία παίρνει σαν είσοδο αρκετά χαρακτηριστικά στιγμιοτύπου χρήστη και υπολογίζει το αρχικό προφίλ χρήστη, ζυγίζοντας το $w_j^u(t)$. Οι τιμές του βάρους αποθηκεύονται στο HDFS για την επόμενη φάση, και διοχετεύονται σε μια δεύτερη ροή MapReduce που συγκεντρώνει αυτές τις τιμές και στη συνέχεια αποθηκεύει την συνολική της αξία στην επόμενη φάση.

Η τελική φάση, έχει μια ροή MapReduce, την calcUserProfileTermWeights, η οποία παίρνει όλες τις ενδιάμεσες τιμές από τις προηγούμενες φάσεις και τους όρους του προφίλ χρήστη $w^u_{j-1}(t)$ που είναι αποθηκευμένα στο HDFS από την προηγούμενη χρονική στιγμή j-1 και εξάγει για κάθε όρο του προφίλ χρήστη τον τελικό όρο KL ζυγισμένο σύμφωνα με την εξίσωση (6).



Algorithm 1: User profiling

Input: Log records: $[u, d]$, Document contents: $[d, text]$
Output: Userprofile weights: $[(u, t), \tilde{w}_j^u(t)]$

Phase (1) [*count the number of documents on each user snapshot* $D_j(u)$ *and calculate intermediate community snapshot* D_j *features*]
begin
  MapReduce flow: `countUserDocs`
  begin
    `Map`$[u, d] \rightarrow [u, 1]$
    `Reduce`$[u, \{1\}] \rightarrow$ `Store`$[u, |D_j(u)|]$
  MapReduce flow: `calcCommunityFeatures`
  begin
    `Map`$[d, text] \rightarrow [t, (d, 1)]$
    `Reduce`$[t, \{(d, 1)\}] \rightarrow$ `Store`$[(t, d), tf(t, d)]$
    `Map`$[(t, d), (tf(t, d), |D_j|)] \rightarrow [t, (tf(t, d), |D_j|, 1)]$
    `Reduce`$[t, \{(tf(t, d), |D_j|, 1)\}] \rightarrow$
    `Store`$[t, (\overline{tf}(t, D_j)cdf(t, D_j), idf(t, D_j))]$
    `Map`$[t, (\overline{tf}(t, D_j)cdf(t, D_j))] \rightarrow$
    $[norm, (\overline{tf}(t, D_j)cdf(t, D_j))]$
    `Reduce`$[norm, \{(\overline{tf}(t, D_j)cdf(t, D_j))\}] \rightarrow$ `Store`$[N_j]$

Phase (2) [*calculate community snapshot* D_j *marginal term distribution and the initial user profile term weights*]
begin
  MapReduce flow: `calcComSnapshotProb`
  begin
    `Map`$[t, (\overline{tf}(t, D_j)cdf(t, D_j), N_j)] \rightarrow [t, P(t|D_j)]$
    `Reduce`$[t, P(t|D_j)] \rightarrow$ `Store`$[t, P(t|D_j)]$
  MapReduce flow: `calcInitialTermWeights`
  begin
    `Map`$[(u, t, d), (tf(t, d), |D_j(u)|, idf(t, D_j))] \rightarrow$
    $[(u, t), (tf(t, d), |D_j|, idf(t, D_j), 1)]$
    `Reduce`$[(u, t), \{(tf(t, d), |D_j|, idf(t, D_j), 1)\}] \rightarrow$
    `Store`$[(u, t), w_j^u(t)]$
    `Map`$[(u, t), w_j^u(t)] \rightarrow [u, w_j^u(t)]$
    `Reduce`$[u, \{w_j^u(t)\}] \rightarrow$ `Store`$[u, \sum_{t \in V} w_j^u(t)]$

Phase (3) [*calculate user snapshot* $D_j(u)$ *marginal term distribution and weight user profile terms using the KL method*]
begin
  MapReduce flow: `calcUserProfileTermWeights`
  begin
    `Map`$[(u, t), (P(t|D_j), w_j^u(t), \sum_{t \in V} w_j^u(t), \tilde{w}_{j-1}^u(t), \alpha_j)] \rightarrow$
    $[(u, t), \tilde{w}_j^u(t)]$
    `Reduce`$[(u, t), \tilde{w}_j^u(t)] \rightarrow$ `Store`$[(u, t), \tilde{w}_j^u(t)]$

Πηγή:M.Shmueli-Scheuer,H.Roitman,D.Carmel,Y.Mass,D.Konopnicki,Extracting User Profiles from Large Scale
Atbrox.com/about

5.6 Σύνολα δεδομένων και πειράματα

Αρχικά περιγράφουμε τα σύνολα των δεδομένων ακολουθούμενα από αξιολόγηση της ποιότητας και ανάλυση κλιμάκωσης του προτεινόμενου πλαισίου. Χρησιμοποιήσαμε δύο σύνολα δεδομένων πραγματικού κόσμου, δεδομένα που έχουν εξαχθεί από το Open Directory Project (ODP) και δεδομένα που έχουν εξαχθεί από blogs, όπως το blogger.com. Τα δεδομένα ODP χρησιμοποιήθηκαν για την ποιοτική ανάλυση και εξομοίωση του περιεχομένου. Επιλέξαμε τυχαία 100 διαφορετικές κατηγορίες από την ιεραρχία ODP. Για κάθε κατηγορία επιλέξαμε τυχαία περισσότερα από 100 έγγραφα με αποτέλεσμα μια συλλογή από 10.000 έγγραφα.

Για την ανάλυση κλιμάκωσης χρησιμοποιήσαμε τα δεδομένα των blog για να προσομοιώσουμε τις αναζητήσεις εγγράφων του χρήστη με τον πάροχο περιεχομένου(σε αυτήν την περίπτωση blogger.com). Σε αυτό το σύνολο δεδομένων, κάθε blog είναι μια συλλογή αναρτήσεων όπου κάθε ανάρτηση μπορεί να έχει προσκολλημένα σε αυτή κάποια σχόλια. Οι χρήστες συνήθως δεν διαβάζουν όλες τις αναρτήσεις του blog, αλλά μάλλον λίγες αναρτήσεις από αυτό. Έτσι θεωρούμε κάθε ανάρτηση με τα σχόλιά της σαν ένα έγγραφο. Επιπλέον κάθε ανάρτηση συσχετίζεται με ένα μοναδικό user id(τον συγγραφέα ανάρτησης). Επιπρόσθετα, κάθε σχόλιο σε μια ανάρτηση συσχετίζεται με κάποιον χρήστη(σχολιαστής). Έτσι θεωρούμε τον συγγραφέα της

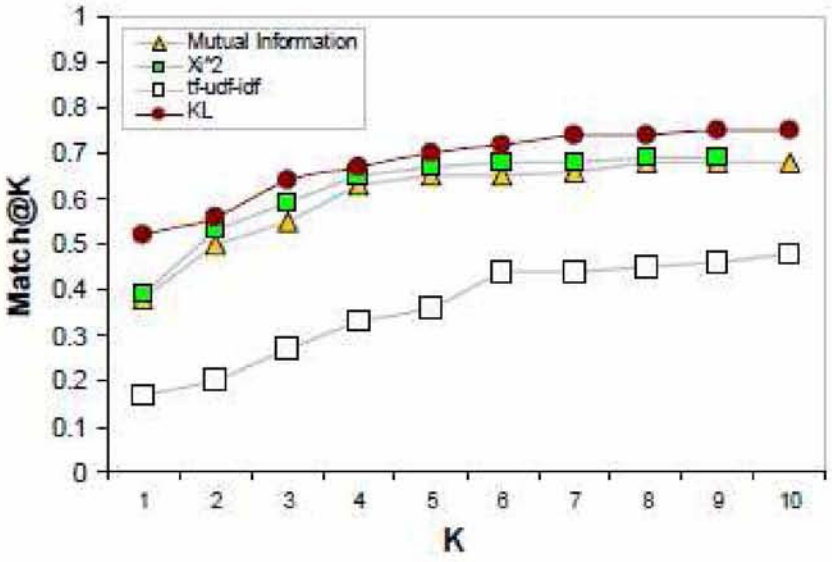
ανάρτησης και το σύνολο των σχολιαστών ως χρήστες που σχετίζονται με αυτήν την ανάρτηση. Επιπλέον το περιεχόμενο της ανάρτησης προκύπτει από το συνδυασμό της αυθεντικής ανάρτησης μαζί με τα σχόλιά της. Ανιχνεύσαμε 973,518 αναρτήσεις blog από τον Μάρτιο του 2007 μέχρι τον Ιανουάριο του 2009 με μια συλλογή 5.45 GB.

Τα πειράματα πραγματοποιήθηκαν χρησιμοποιώντας 4 κόμβους(κάθε μηχανή με 4GB RAM, 60GB HD, και τέσσερις πυρήνες)και κάθε κόμβος λειτουργικό σύστημα Linux-ubuntu και Hadoop έκδοση 0.20.1.

5.7 Ανάλυση ποιότητας

Ξεκινάμε με ανάλυση ποιότητας του προτεινόμενου μοντέλου προφίλ χρήστη, που βασίζεται στην μέθοδο επιλογής KL χαρακτηριστικών, όπως περιγράφηκε παραπάνω.

Συγκρίναμε την μέθοδο KL με δύο άλλες μεθόδους επιλογής χαρακτηριστικών την MI(mutual information) και χ^2 [5]. Συγκρίναμε την μέθοδο KL με την απλή προσέγγιση η οποία σταθμίζει τους όρους προφίλ χρήστη σύμφωνα με την εξίσωση (2)(που υποδηλώνεται ως tf-udf-idf).Ακολουθήσαμε το πλαίσιο αξιολόγησης του Carmel[1] και αξιολογήσαμε την επίδοση επιλογής χαρακτηριστικών χρησιμοποιώντας το μέτρο Match@K,το οποίο μετράει την πιθανότητα να πάρουμε τουλάχιστον ένα σωστό χαρακτηριστικό από το σύνολο τον κορυφαίων K προτεινόμενων χαρακτηριστικών(ετικέτες).Το επόμενο σχήμα δείχνει το Match@K που έχει παρατηρηθεί για τις διαφορετικές μεθόδους επιλογής χαρακτηριστικών. Παρατηρούμε ότι η μέθοδος KL έχει υψηλότερες επιδόσεις από τις άλλες μεθόδους. Για παράδειγμα σε τουλάχιστον 50% των περιπτώσεων, η πρώτη ετικέτα που προτάθηκε από την μέθοδο KL ήταν σωστή(τουλ.20%καλύτερη από άλλες μεθόδους).



Match@k για τις μεθόδους επιλογής χαρακτηριστικών.
Πηγή:M.Shmueli-Scheuer,H.Roitman,D.Carmel,Y.Mass,D.Konopnicki,Extracting User Profiles from Large Scale
Atbrox.com/about

5.8 Ανάλυση κλιμάκωσης

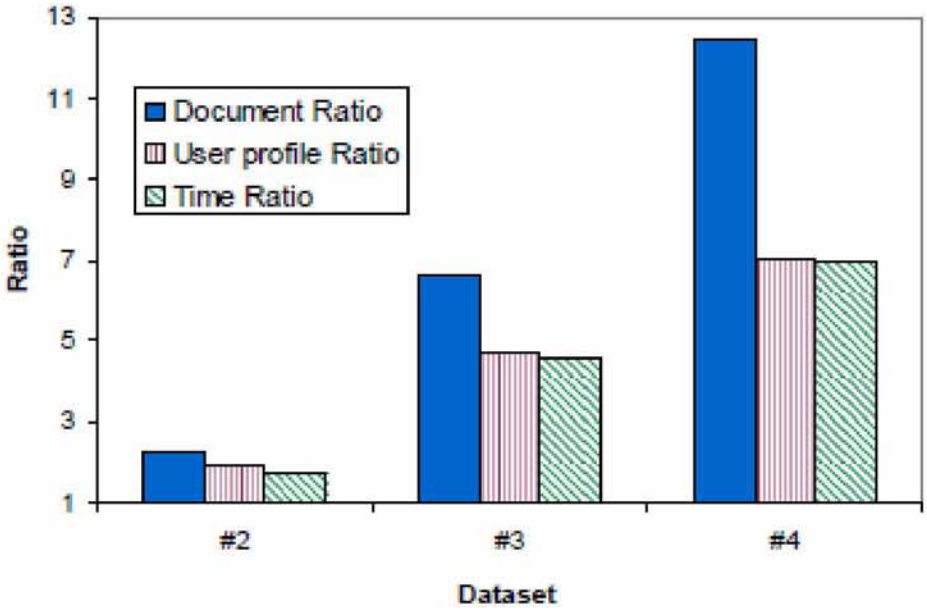
Στη συνέχεια αναλύουμε την κλιμάκωση του πλαισίου μας, σε σχέση με τις διάφορες ρυθμίσεις παραμέτρων. Εντοπίζουμε δύο κύριους παράγοντες που μπορούν να επηρεάσουν την επίδοση του αλγόριθμου 1, δηλαδή τον αριθμό των προφίλ χρηστών που πρέπει να διατηρήσουμε και τον αριθμό των εγγράφων που σχετίζονται με τους χρήστες στο σύστημα(μέγεθος δεδομένων).

5.8.1 Αριθμός προφίλ χρηστών

Πρώτα αναλύουμε την απόδοση σε σχέση με τον αριθμό των προφίλ χρηστών που χρειάζεται να διατηρεί το πλαίσιο. Αξίζει να σημειωθεί ότι όταν ο αριθμός των προφίλ χρηστών αυξάνεται, περιμένουμε τον αριθμό τον εγγράφων να αυξηθεί και αυτός. Μεταβάλλαμε τον αριθμού των προφίλ χρηστών μεταξύ 20.000 και 120.000 και λάβαμε σαν αποτέλεσμα από 40.000 μέχρι 500.000 έγγραφα με συνολικό μέγεθος δεδομένων από 250 MB έως 3Gb. Χρησιμοποιήσαμε cluster 4 κόμβων και μετρήσαμε τον συνολικό χρόνο εκτέλεσης (σε λεπτά) της εργασίας σκιαγράφησης των προφίλ. Ο παρακάτω πίνακας συνοψίζει τις ρυθμίσεις και τον χρόνο εκτέλεσης που προκύπτει. Η γραφική παράσταση που ακολουθεί απεικονίζει τις σχετικές αναλογίες σε σχέση με το μικρότερο σύνολο δεδομένων(δηλαδή την πρώτη γραμμή, #1). Για παράδειγμα τα προφίλ χρηστών, τα δεδομένα και οι αναλογίες χρόνου για το σύνολο δεδομένων #3 διαφέρουν σε σχέση με το σύνολο δεδομένων #1 κατά 4.68,6.60 και 4.55 αντίστοιχα. Αυτό που παρατηρούμε είναι ότι η αναλογία χρόνων εκτέλεσης συσχετίζεται με την αναλογία προφίλ χρηστών παρά με την αναλογία δεδομένων το οποίο σημαίνει ότι ο αριθμός των προφίλ χρηστών και όχι ο αριθμός των εγγράφων επηρεάζουν την επίδοση του αλγόριθμου.

Dataset	# user profiles	# documents	Runtime (min.)
#1	17,786	39,596	39.21
#2	34,186	88,563	67.76
#3	83,344	261,509	170.71
#4	124,782	493,490	272.39

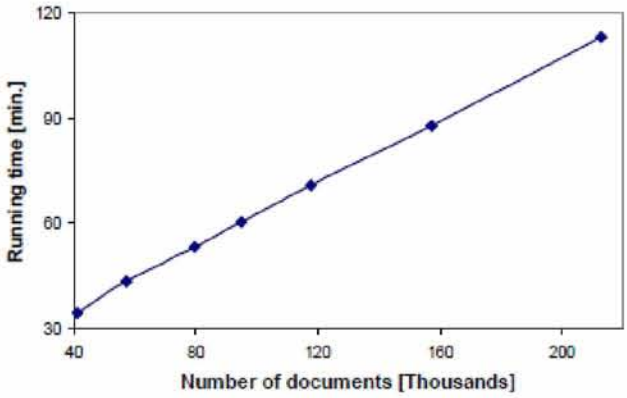
runtime performance
Πηγή:M.Shmueli-Scheuer,H.Roitman,D.Carmel,Y.Mass,D.Konopnicki,Extracting User Profiles from Large Scale
Atbrox.com/about



Αναλογία συνόλων δεδομένων
M.Shmueli-Scheuer,H.Roitman,D.Carmel,Y.Mass,D.Konopnicki,Extracting User Profiles from Large Scale
Atbrox.com/about

5.8.2 Μέγεθος δεδομένων

Τέλος, αναλύουμε την κλιμάκωση του πλαισίου μας σε σχέση με το μέγεθος των δεδομένων εφόσον λάβουμε υπόψιν μας τον αριθμό των εγγράφων που χρειάζονται να επεξεργαστούν από το πλαίσιο. Επιλέξαμε ένα υποσύνολο από περίπου 180.000 χρήστες που υπήρχαν στο σύνολο δεδομένων κατά την διάρκεια των μηνών Μάρτιος - Απρίλιος 2007 και εξάγαμε τα σχετιζόμενα έγγραφά τους για τους μήνες Μάρτιο - Αύγουστο του 2007,έτσι ώστε ο συνολικός αριθμός των εγγράφων να ποικίλει μεταξύ 41.227-157.330 με αποτέλεσμα ένα συνολικό μέγεθος δεδομένων 250 MB έως 1.2 Gb αντίστοιχα. Χρησιμοποιήσαμε το cluster των 4 κόμβων και μετρήσαμε τον συνολικό χρόνο εκτέλεσης της εργασίας. Αυτό που παρατηρήσαμε ήταν οτι ο χρόνος εκτέλεσης



Απόδοση του χρόνου εκτέλεσης με τηναύξηση του μεγέθους των δεδομένων
Πηγή:M.Shmueli-Scheuer,H.Roitman,D.Carmel,Y.Mass,D.Konopnicki,Extracting User Profiles from Large Scale.Atbrox.com/about

αυξάνεται γραμμικά με την αύξηση του μεγέθους των δεδομένων. Ακόμη, ενώ το μέγεθος των δεδομένων επηρεάζει γραμμικά το χρόνο εκτέλεσης, αυτή η επίδραση είναι αμελητέα όταν ο αριθμός των προφίλ χρηστών αυξάνεται.

5.9 Συμπεράσματα

Προτείναμε μια λύση κλιμακούμενης σκιαγράφησης του χρήστη που εφαρμόζεται στο πλαίσιο του Hadoop MapReduce.

Βιβλιογραφία

[1] D. Carmel, H. Roitman, and N. Zwerdling. Enhancing cluster labeling using wikipedia. In SIGIR '09, pages 139–146, New York, NY, USA, 2009. ACM.

[2] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? In SIGIR '06, pages 390–397. ACM Press, 2006.

[3] L. Chen and K. Sycara. Webmate: a personal agent for browsing and searching. In AGENTS '98, New York, NY, USA, 1998. ACM.

[4] S. Kullback and R. A. Leibler. On information and sufficiency. The Annals of Mathematical Statistics, 22(1):79–86, 1951

[5] C. D. Manning, P. Raghavan, and H. Schutze. Introduction to Information Retrieval. Cambridge University Press, 2008.

[6] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In WWW, pages 675–684, 2004

M. Shmueli-Scheuer, H. Roitman, D. Carmel, Y. Mass, D. Konopnicki, "Extracting User Profile from Large Scale Data", 2010.

6. Κατανεμημένος αλγόριθμος για τον υπολογισμό τυπικών εννοιών

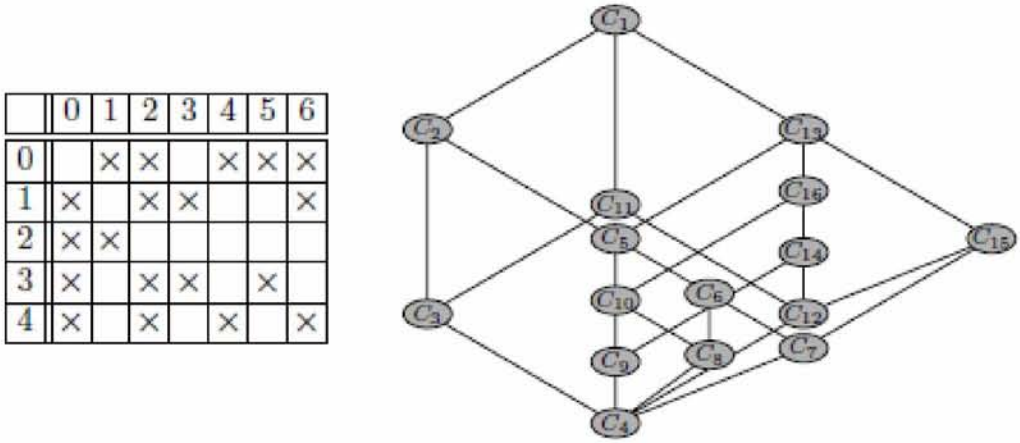
Η αναζήτηση για ενδιαφέροντα πρότυπα σε δυαδικά σύνολα παίζει σημαντικό ρόλο στην εξόρυξη δεδομένων και ιδίως στην ανάλυση τυπικών εννοιών. Προτάθηκαν διάφοροι αλγόριθμοι για τον υπολογισμό προτύπων που αναπαριστώνται με μέγιστα ορθογώνια σε δυαδικά σύνολα αλλά το μεγαλύτερό τους μειονέκτημα είναι η υπολογιστική τους περιπλοκότητα που περιορίζει την εφαρμογή σε σχετικά μικρά σύνολα δεδομένων. Η προσέγγισή που παρουσιάζουμε είναι μοναδική ανάμεσα σε άλλες στο γεγονός ότι χρησιμοποιούμε το πλαίσιο MapReduce το οποίο παραδοσιακά χρησιμοποιείται για αναζητήσεις και επερωτήσεις σε μεγάλες συλλογές δεδομένων. Θα παρουσιάσουμε μια πρώιμη μελέτη και μια απόδειξη της έννοιας του πως το πλαίσιο MapReduce μπορεί να χρησιμοποιηθεί για συγκεκριμένες εργασίες εξόρυξης δεδομένων.

Επικεντρωνόμαστε στο να εξάγουμε ορθογώνια πρότυπα ,οι επονομαζόμενες τυπικές έννοιες, σε δυαδικά σχεσιακά δεδομένα. Ο Belohlavec και ο Vychodil έδειξαν [1] ότι τα μέγιστα ορθογώνια μπορούν να χρησιμοποιηθούν για να βρεθεί βέλτιστη παραγοντοποίηση των συνόλων boolean. Το να βρούμε μέγιστα ορθογώνια σε πίνακες δεδομένων είναι επομένως μια σημαντική εργασία. Ο αλγόριθμος που προτείνουμε ίσως βοηθήσει να ξεπεραστούν τα προβλήματα με το να παράγουμε όλες τις τυπικές έννοιες από μεγάλα σύνολα δεδομένων. Γενικά, το πρόβλημα καταγραφής όλων των τυπικών εννοιών είναι #P-complete[14].Ευτυχώς αν τα δεδομένα εισόδου είναι αραιά ,μπορούμε να έχουμε σύνολα από όλες τις τυπικές έννοιες σε λογικό χρόνο. Ακόμα η καταγραφή όλων των τυπικών εννοιών απαιτεί χρόνο και χώρο και επομένως υπάρχει η ανάγκη να παρέχουμε κλιμακωτούς κατανεμημένους αλγορίθμους ,το οποίο θα μας βοηθήσει να διανύμουμε την επιβάρυνση σε ένα μεγάλο αριθμό υπολογιστικών κόμβων χαμηλού κόστους.

6.1 Ανάλυση τυπικών εννοιών(formal concept analysis)

Σε αυτό το τμήμα ανακαλούμε βασικές έννοιες της ανάλυσης τυπικών εννοιών (FCA) .Οι περισσότερες λεπτομέρειες μπορούν να βρεθούν στα μονογραφήματα [7][3].Η ανάλυση τυπικών εννοιών ασχολείται με δυαδικούς πίνακες δεδομένων που περιγράφουν την σχέση μεταξύ αντικειμένων-χαρακτηριστικών. Τα δεδομένα εισόδου τα οποία μας ενδιαφέρουν παίρνουν την μορφή ενός δισδιάστατου πίνακα δεδομένων με σειρές που αντιστοιχούν στα αντικείμενα ,στήλες που αντιστοιχούν στα χαρακτηριστικά, και καταχωρήσεις στον πίνακα που είναι σταυροί (ή 1) και κενά(ή 0),που δείχνουν την παρουσία/απουσία χαρακτηριστικών : Ένας πίνακας τσεκαριστεί με x στην τομή της γραμμής που αντιστοιχεί στο αντικείμενο x και της στήλης που αντιστοιχεί στο χαρακτηριστικό y αν και μόνο αν "το αντικείμενο x έχει χαρακτηριστικά y"("το χαρακτηριστικό y υπάρχει στο αντικείμενο x").Δοθέντος ενός πίνακα δεδομένων ,ευχόμαστε να βρούμε όλους τους μέγιστους υποπίνακες(matrices) από x που υπάρχουν στον πίνακα. Ένα παράδειγμα από έναν τέτοιο πίνακα δεδομένων απεικονίζεται στην εικόνα 1(αριστερά),τον οποίο μπορούμε να τον δούμε σαν μια δυαδική σχέση $I \subseteq X \times Y$

τέτοια ώστε <x, y> ∈I ανν αντικείμενο x έχει χαρακτηριστικό y. Στο FCA το I συνήθως ονομάζεται τυπικό πλαίσιο[7].



	0	1	2	3	4	5	6
0		×	×		×	×	×
1	×		×	×			×
2	×	×					
3	×		×	×		×	
4	×		×		×		×

Εικόνα 1:Πίνακας δεδομένων(αριστερα),πλέγμα που προκύπτει από το αριστερό κομμάτι(δεξιά).
Πηγή:P.Krajca,V.Vychodil,Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework
atbrox.com/about

Θα χρησιμοποιήσουμε ένα σύνολο X={0,1,....,m} αντικειμένων και ένα σύνολο Y={0,1,2,...,n}χαρακτηριστικών. Δεν υπάρχει κίνδυνος σύγχυσης αντικειμένων με χαρακτηριστικά επειδή δεν αναμιγνύουμε στοιχεία από τα σύνολα X και Y ποτέ. Κάθε περιεχόμενο I $\subseteq$ X ×Y εισάγει ένα ζευγάρι τελεστών ↑ και ↓ για κάθε A $\subseteq$ X και B $\subseteq$ Y ακολούθως:

- $A^{\uparrow}$ = **{y ∈Y | για κάθε x ∈A: x, y ∈I}** **(1)**
- $B^{\downarrow}$ = **{x ∈X | για κάθε y ∈B: x, y ∈I}** **(2)**

Τελεστές ↑: $2^X \rightarrow 2^Y$ και ↓: $2^Y \rightarrow 2^X$ ορισμένοι στις σχέσεις (1) και (2) ονομαζόμενη σύνδεση Galois[7].Από τον ορισμό (1) ,το A↑ είναι ένα σύνολο όλων των γνωρισμάτων που μοιράζονται όλα τα αντικείμενα από το A και από τον (2) ,B ↓ είναι το σύνολο όλων των αντικειμένων που μοιράζονται όλα τα χαρακτηριστικά από το B. Ένα ζευγάρι <A,B> όπου A $\subseteq$ X, B $\subseteq$ Y , $A^{\uparrow}$ = B και $B^{\downarrow}$ = A αποκαλείται τυπική έννοια (όπου I $\subseteq$ X × Y).Οι τυπικές έννοιες μπορούν να θεωρηθούν σαν συστάδες κρυμμένες στα δεδομένα. Δηλαδή, αν <A,B> είναι μια τυπική έννοια ,όπου A() είναι ένα σύνολο από όλα τα αντικείμενα που μοιράζονται όλα τα χαρακτηριστικά του B και αντιστρόφως το B είναι το σύνολο όλων των γνωρισμάτων που μοιράζονται από όλα τα αντικείμενα του A. Από τεχνικής πλευράς οι τυπικές έννοιες είναι σταθερά σημεία στην σύνδεση Galois <↑, ↓> που επηρεάζονται από το επίσημο πλαίσιο. Τυπικές έννοιες στο I $\subseteq$X × Y αντιστοιχούν στα μέγιστα ορθογώνια στο πλαίσιο I. Πιο αναλυτικά ,κάθε <A,B>∈ 2^X × 2^Y τέτοιο ώστε το A × B $\subseteq$I θα ονομάζεται ένα ορθογώνιο στο I. Το ορθογώνιο A,B στο I είναι το μέγιστο δυνατό εάν για οποιοδήποτε ορθογώνιο <A',B'> στο I τέτοιο ώστε A ×

B ⊆ A'*B', θα έχουμε ότι A=A' και B=B'. Έχουμε ότι <A,B>∈ $2^X \times 2^Y$, το μέγιστο ορθογώνιο στο I ανν το A↑= B και B↓=A ,δηλαδή τα μέγιστα ορθογώνια ισούνται με τις τυπικές έννοιες. Επομένως ,τα μέγιστα ορθογώνια μας δίνουν μια εναλλακτική ερμηνεία των τυπικών εννοιών.

Έστω B(X, Y, I) δηλώνει το σύνολο όλων των τυπικών εννοιών στο I ⊆X × Y. Το σύνολο B(X, Y, I) περιλαμβάνει μια μερική κατάταξη (<=) μοντελοποιώντας ιεραρχικά τα υπερσύνολα :

<A1,B1> ≤ <A2,B2> ανν A1 ⊆A2 (ή, ισοδύναμα, ανν B2 ⊆B1) (3).

Αν <A1,B1> ≤ <A2,B2> ,τότε το A1,B1 είναι ένα υποσύνολο του <A2,B2>.Το σύνολο B(X, Y, I), μαζί με το ≤ μας δίνουν ένα πλήρες πλέγμα ,η δομή του οποίου έχει περιγραφεί από το βασικό θεώρημα της ανάλυσης τυπικής έννοιας.[7]. Η παραπάνω περιγραφές εννοιών διασαφηνίζονται στο παρακάτω παράδειγμα:

Παράδειγμα1:Θεωρείστε ένα τυπικό περιβάλλον I ⊆ X×Y και το οποίο αντιστοιχίζεται στον πίνακα δεδομένων στην Εικόνα 1(μόνο το αριστερό τμήμα)

C1 = {0, 1, 2, 3, 4}, {},
C2 = {1, 2, 3, 4}, {0},
C3 = {2}, {0, 1},
C4 = {}, {0, 1, 2, 3, 4, 5, 6},
C5 = {1, 3, 4}, {0, 2},
C6 = {1, 3}, {0, 2, 3},
C7 = {3}, {0, 2, 3, 5},
C8 = {1}, {0, 2, 3, 6},
C9 = {4}, {0, 2, 4, 6},
C10 = {1, 4}, {0, 2, 6},
C11 = {0, 2}, {1},
C12 = {0}, {1, 2, 4, 5, 6},
C13 = {0, 1, 3, 4}, {2},
C14 = {0, 4}, {2, 4, 6},
C15 = {0, 3}, {2, 5},
C16 = {0, 1, 4}, {2, 6}.

B(X,Y,I)=(C1,…C16).Αν εφαρμόσουμε στο B(X,Y,I) την (3).θα πάρουμε το δεξί μέρος της εικόνας 1.

6.2Επεξεργασία δεδομένων χρησιμοποιώντας την προσέγγιση MapReduce

Οι εφαρμογές κοινών κατανεμημένων αλγορίθμων συνήθως αποτελούνται από παραλλαγές περισσότερο ή λιγότερο γνωστών αλγορίθμων που κατανέμουν συγκεκριμένα τμήματα των δεδομένων για επεξεργασία σε άλλους υπολογιστές. Αυτή η προσέγγιση είναι κατανοητή για τους προγραμματιστές αλλά επιφέρει διάφορα θέματα. Για αυτόν τον λόγο, θα χρησιμοποιήσουμε το πλαίσιο Map-Reduce για την επεξεργασία πολλών δεδομένων σε κατανεμημένα δίκτυα, το οποίο προτείνεται στο[4].Βασίζεται στις δυο βασικές λειτουργίες Map και Reduce που εφαρμόζονται στα δεδομένα. Αυτές οι λειτουργίες έχουν δώσει το όνομα του πλαισίου ---map-reduce πλαίσιο ή(πιο σύντομα ένα πλαίσιο M/R).Αυτή η προσέγγιση για την επεξεργασία δεδομένων είχε αρχικά αναπτυχθεί από την google για τα κέντρα δεδομένων της αλλά αποδείχθηκε πολύ πρακτική και αργότερα προσαρμόστηκε από άλλες εταιρείες λογισμικού που ενδιαφέρονταν στο να αποθηκεύουν και να αναζητούν μεγάλου όγκου δεδομένων.

6.3Ανασκόπιση λειτουργίας mapreduce(M/R)

Τα δεδομένα στο M/R παρουσιάζονται στην μορφή ζευγαριών <key-value> .Στο πρώτο βήμα του υπολογισμού διαβάζει δεδομένα εισόδου και προαιρετικά τα μετατρέπει στα επιθυμητά ζευγάρια κλειδί-τιμή. Στο δεύτερο βήμα δηλαδή στη φάση map ,μια συνάρτηση f εφαρμόζεται σε κάθε ζεύγος<k,v> και επιστρέφει ένα πολυσύνολο νέων ζευγαριών key,value π.χ. f(k, v) = {k1, v1, . . . , kn, vn}. Σημειώστε ότι υπάρχει μεγάλη ομοιότητα με την συνάρτηση map που υπάρχει σε πολλές γλώσσες προγραμματισμού(π.χ.LISP,Python και Scheme). Σε αντίθεση με τη συνηθισμένη map, η συνάρτηση f ίσως επιστρέψει αυθαίρετο αριθμό αποτελεσμάτων και όλα συλλέγονται κατά την διάρκεια της φάσης map. Επομένως, στην φάση reduce,όλα τα ζευγάρια που έχουν παραχθεί στο πλοηγούμενο βήμα ομαδοποιούνται από τα κλειδιά τους και οι τιμές τους μειώνονται από μια συνάρτηση g({k, v1, k, v2, . . . , k, vn}) = <k, v>. Το ακόλουθο παράδειγμα δείχνει πως το πλαίσιο M/R μπορεί να χρησιμοποιηθεί για να εκτελέσει έναν υπολογισμό ο οποίος ίσως να εμφανιστεί στην ανάκτηση πληροφοριών.

Παράδειγμα 2 : Ας υποθέσουμε ότι θέλουμε να υπολογίσουμε συχνότητες γραμμάτων σε ένα κείμενο που αποτελείται από τρεις λέξεις -alice, barbara και carol.Τώρα ας υποθέσουμε ότι η συνάρτηση f δέχεται μια λέξη και επιστρέφει ένα πολυσύνολο όπου κάθε γράμμα στην λέξη αντιπροσωπεύεται σαν ένα ζεύγος(γράμμα, 1).Η φάση map θα παράξει τα ακόλουθα αποτελέσματα :

f(alice) = {a, 1, l, 1, i, 1, c, 1, e, 1},
f(barbara) = {b, 1, a, 1, r, 1, b, 1, a, 1, r, 1, a, 1},
f(carol) = {c, 1, a, 1, r, 1, o, 1, l, 1}.

Στην φάση reduce ομαδοποιούμε όλα τα ζευγάρια με το κλειδί και η συνάρτηση g αθροίζει όλα τα 1 στα ζευγάρια key-value ακολούθως:

g({a, 1, a, 1, a, 1, a, 1, a, 1}) = a, 5,
g({b, 1, b, 1}) = b, 2,
g({c, 1, c, 1}) = c, 2,
g({e, 1}) = e, 1,
g({l, 1, l, 1}) = l, 2,
g({i, 1}) = i, 1,
g({o, 1}) = o, 1,
g({r, 1, r, 1, r, 1}) = r, 3.

Μπορούμε να δούμε από το προηγούμενο παράδειγμα και το σκίτσο του αλγόριθμου ότι εφόσον οι συναρτήσεις f και g εφαρμόζονται μόνο σε συγκεκριμένα ζευγάρια, είναι πιθανό να διανύμουμε εύκολα τον υπολογισμό σε διάφορους υπολογιστές. Για χάρη της ολοκλήρωσης πρέπει να τονίσουμε ότι οι συναρτήσεις f και g που χρησιμοποιούνται για να κάνουν map και να αθροίσουν τιμές, πρέπει πάντα να εφαρμόζονται από διαδικασίες(π.χ. υπολογιστικές συναρτήσεις) που δεν έχουν

ανεπιθύμητες επιπτώσεις. Αυτό σημαίνει ότι τα αποτελέσματα f και g βασίζονται μόνο σε δεδομένα ορίσματα π.χ. οι διαδικασίες που υπολογίζουν τα αποτελέσματα των f και g συμπεριφέρονται ως maps.

6.4 Αλγόριθμος

Οι πιο κοινοί αλγόριθμοι για τυπικές έννοιες υπολογισμού περιλαμβάνουν τον αλγόριθμο του Ganter[6],του Linding[18] και Berry[2].Σε αυτό το σημείο παρουσιάζουμε μια περίληψη του παράλληλου αλγόριθμου PCBO που είχαμε προτείνει στο [12],[13].Στη συνέχεια παρουσιάζουμε μια κατανεμημένη παραλλαγή του PCBO βασισμένη στο MapReduce. Επίσης, μια αποτίμηση και σύγκριση των αλγορίθμων για FCA βρίσκεται στο [17].Ο κατανεμημένος αλγόριθμος είναι μια προσαρμογή του αλγόριθμου Kuznetsov's Close-by-One και της παράλληλης παραλλαγής PCBO.Ο CBO μπορεί να τυποποιηθεί από μια αναδρομική διαδικασία GenerateFrom(A,B, y), η οποία καταγράφει όλα τα τυπικά θέματα χρησιμοποιώντας μια αναζήτηση κατά βάθος στο διάστημα που βρίσκονται όλες οι τυπικές έννοιες. Η διαδικασία δέχεται μια τυπική έννοια <A,B>,(δηλαδή μια αρχική τυπική έννοια) και ένα χαρακτηριστικό $y \in Y$ ως ορίσματα. Η διαδικασία μειώνεται αναδρομικά κατά μήκος του διαστήματος των τυπικών εννοιών ξεκινώντας με την τυπική έννοια <A,B>.

Όταν καλείται η GenerateFrom με <A,B> και $y \in Y$ αρχικά επεξεργάζεται τα A,B (για παράδειγμα το τυπώνει στην οθόνη ή το αποθηκεύει σε μια δομή δεδομένων) και μετά ελέγχει την τερματική κατάσταση. Ο υπολογισμός σταματάει είτε όταν το <A,B> ισούται $(Y\downarrow, Y)$ είτε $y>n$(δηλαδή δεν απομένουν άλλα χαρακτηριστικά για επεξεργασία).Διαφορετικά η διαδικασία συνεχίζεται για όλα τα χαρακτηριστικά $j \in Y$ τέτοια ώστε $j \geq y$ και τα οποία δεν υπάρχουν στο B.Για κάθε $j \in Y$ που έχει αυτές τις ιδιότητες ,υπολογίζεται ένα καινούριο ζευγάρι $<C,D> \in 2^X \times 2^y$ τέτοιο ώστε:

$$\mathbf{<C,D>=<A\cap \{j\}^{\downarrow}\ ,\ (A\cap \{j\}^{\downarrow})^{\uparrow}>} \quad \mathbf{(4)}$$

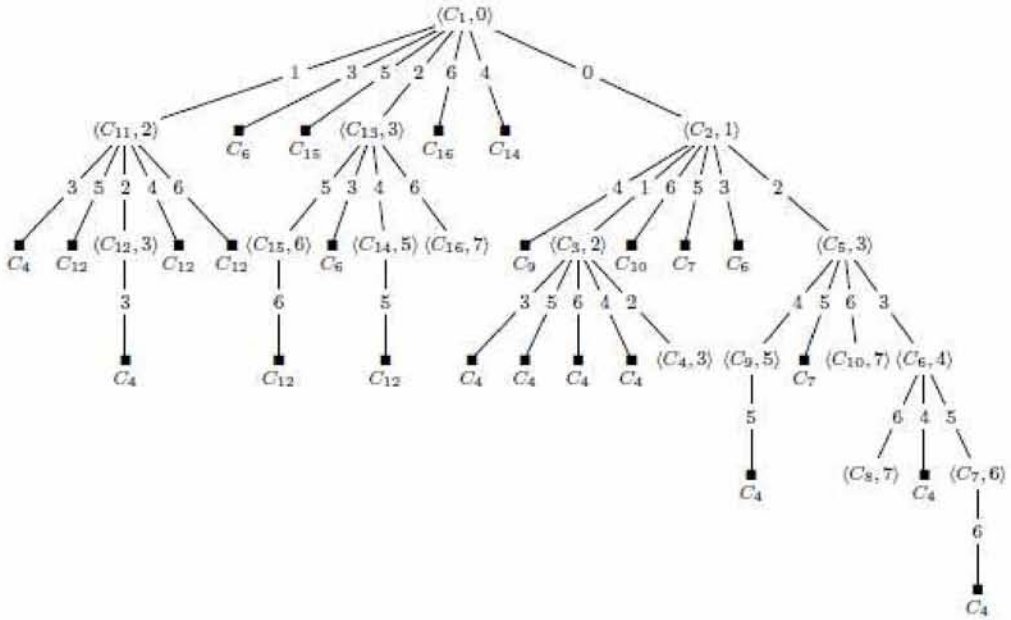
Το ζευγάρι <C,D> είναι πάντα ένα τυπικό ζήτημα [12] έτσι ώστε $B \subset D$. Αφού πάρουμε το <C,D>,ο αλγόριθμος ελέγχει πότε θα συνεχίσει με το C,D καλώντας αναδρομικά την GenerateFrom ή πότε το <C,D> θα αγνοηθεί. Το τέστ (όπως ονομάζεται το τεστ κανονικότητας) βασίζεται στην σύγκριση $B \cap Yj = D \cap Yj$ όπου $Yj \subseteq Y$ και ορίζεται ακολούθως:

$$\mathbf{Yj = \{y \in Y\ |y < j\}.} \quad \mathbf{(5)}$$

Ο ρόλος του τεστ κανονικότητας είναι να προλάβει τον υπολογισμό του ίδιου τυπικού θέματος πολλές φορές. Η GenerateFrom υπολογίζει τα τυπικά θέματα με μια μοναδική σειρά, το οποίο εξασφαλίζει ότι κάθε τυπικό θέμα έχει επεξεργαστεί ακριβώς μια φορά[13].

Σημείωση 1:Η αναδρομικές κλήσεις της GenerateFrom δημιουργούν ένα δέντρο. Το δέντρο ανάλογα με τα δεδομένα της εικόνας 1 ,αναπαριστάται στην εικόνα 2.Η ρίζα αντιστοιχίζεται στην πρώτη κλήση της GenerateFrom($<\varnothing^{\downarrow},\varnothing^{\downarrow\uparrow}>$,0). Κάθε κόμβος με ετικέτα <Ci, k> αντιστοιχεί σε μια κλήση της GenerateFrom όπου το Ci είναι ένα τυπικό θέμα(Δες παράδειγμα 1).Οι κόμβοι που φαίνονται με μαύρα τετραγωνάκια

αναπαριστούνε θέματα που έχουν υπολογιστεί αλλά δεν έχουν επεξεργαστεί διότι το τεστ κανονικότητας απέτυχε. Οι κορυφές του δέντρου έχουν ετικέτα με τον αριθμό των χαρακτηριστικών που χρησιμοποιούνται για να υπολογιστούν τα νέα θέματα. CF[4].



Εικόνα 2:Αναπαράσταση των δεδομένων της εικόνας 1 σε δέντρο για το GenerateFrom(<$\varnothing^{\downarrow}$,$\varnothing^{\downarrow\uparrow}$>,0)
Πηγή:P.Krajca,V.Vychodil,Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework
atbrox.com/about

6.4.1 Προσαρμογή για το πλαίσιο M/R

Στον κατανεμημένο αλγόριθμο θα χρησιμοποιήσουμε την στρατηγική κατά πλάτους αναζήτησης .Μετατροπή του αλγόριθμου από αναζήτηση κατά βάθος (που περιγράφηκε παραπάνω) σε αναζήτηση κατά πλάτος είναι απαραίτητη για την προσαρμογή του CbO στο mapreduce πλαίσιο. Δεδομένου ότι η GenerateFrom εξαρτάται μόνο από τα ορίσματά της, η στρατηγική με βάση την οποία δημιουργούνται θέματα δεν διαδραματίζει σημαντικό ρόλο και μπορεί να αντικατασταθεί από κάποια άλλη. Η αρχική GenerateFrom χωρίζεται σε δυο συναρτήσεις, μια συνάρτηση map που θα ονομάζεται MAP-CONCEPTS και μια συνάρτηση ελαχιστοποίησης που θα την ονομάζουμε REDUCECONCEPTS.Η συνάρτηση map θα ασχοληθεί με την παραγωγή νέων τυπικών διαδικασιών και η συνάρτηση reduce θα ασχοληθεί με την εκτέλεση των τεστ κανονικότητας .Η στρατηγική της αναζήτησης κατά πλάτος είναι ωφέλιμη όσο μας επιτρέπει να υπολογίζουμε τυπικές διαδικασίες σε επίπεδα, όπου κάθε επίπεδο

υπολογίζεται από το προηγούμενό του με συνεχόμενες εφαρμογές των MAP-CONCEPTS και REDUCECONCEPTS στις τυπικές διαδικασίες που υπολογίστηκαν στο προηγούμενο επίπεδο.

Σημείωση 2: Η προσαρμογή για το πλαίσιο mapreduce έχει πολλές πλευρές. Πρώτον αντί να χρησιμοποιούμε μια μοναδική αναδρομική διαδικασία GenerateFrom απασχολούμε δύο συναρτήσεις που δεν είναι αναδρομικές(δηλαδή δεν επιδρά η μια στην άλλη),αλλά χρησιμεύουν ως συναρτήσεις mapping και reduction. Δεύτερον, τα ορίσματα στα MAPCONCEPTS και REDUCECONCEPTS τα οποία με κάποιο τρόπο κωδικοποιούν τα ορίσματα του πρωτότυπου GENERATEFROM θα πρέπει να παρουσιάζονται σαν ζευγάρια key/value έτσι ώστε να μας εξασφαλίζουν τη συμβατότητα με το πλαίσιο mapreduce. Θυμηθείτε πως το mapreduce προϋποθέτει ότι όλες οι τιμές που έχουν επεξεργαστεί από τις MAPCONCEPTS και REDUCECONCEPTS θα είναι της μορφής <key,value>.Για να εξασφαλίσουμε την συμβατότητα είσοδο/εξόδου με το πλαίσιο map-reduce θα κωδικοποιήσουμε τα ορίσματα για τις συναρτήσεις MAPCONCEPTS και REDUCECONCEPTS όπως φαίνεται παρακάτω:
Θεωρούμε ζευγάρια key,value τέτοια ώστε:

-το key είναι μια πλειάδα <B,y> όπου B είναι η πρόθεση της έννοιας <A,B>$\in$B(X, Y, I) και y $\in$Y είναι ένα χαρακτηριστικό.
-το value είναι μια νέα έννοια <C,D>$\in$B(X, Y, I).

Η ακριβής έννοια του ζευγαριού key,value κατά την διάρκεια του υπολογισμού θα γίνει προφανής αργότερα. Ο τρόπος που οι συναρτήσεις MAPCONCEPTS και REDUCECONCEPTS υπολογίζουν όλες τις τυπικές διαδικασίες μπορεί να συνοψιστεί στα παρακάτω βήματα:

Βήμα 1^{ο}: Αρχικά η πρώτη τυπική διαδικασία $<\varnothing^{\downarrow}, \varnothing^{\downarrow\uparrow}>$ υπολογίζεται και καλείται η MAPCONCEPTS με αρχικό ζευγάρι key,value $<<\varnothing^{\downarrow\uparrow},0>,<\varnothing^{\downarrow},\varnothing^{\downarrow\uparrow}>>$ που παράγει ένα πολυσύνολο από καινούρια ζευγάρια key,value τα οποία αναπαριστούν νέες διαδικασίες. Αυτό το πολυσύνολο, μειώνεται μέσω της συνάρτησης REDUCECONCEPTS με τη διαγραφή των ζευγαριών key/value με διαδικασίες που δεν καταφέρνουν να περάσουν τα τεστ κανονικότητας .Αυτά τα δυο βήματα αναπαριστούν την πρώτη επανάληψη.

Βήμα 2^{ο}: Η συνάρτηση MAPCONCEPTS εφαρμόζεται σε κάθε ζευγάρι key,value από την προηγούμενη ν-στη επανάληψη και το αποτέλεσμα ελαχιστοποιείται με την REDUCECONCEPTS. Τα επιστρεφόμενα ζευγάρια key/value περιέχουν τυπικές διαδικασίες που είναι αποθηκευμένες ως αποτέλεσμα της n+1 επανάληψης.

Βήμα 3^{ο}: Αν η (n+1)στη επανάληψη παράγει νέες έννοιες, ο υπολογισμός συνεχίζει με το βήμα 2 για την επόμενη επανάληψη. Διαφορετικά ο υπολογισμός σταματά. Στη συνέχεια, παρέχουμε μια λεπτομερή περιγραφή των MAPCONCEPTS και REDUCECONCEPTS.

6.4.2 Λεπτομέρειες σχετικά με MAPCONCEPTS και REDUCECONCEPTS

Η συνάρτηση mapping περιγράφεται στον αλγόριθμο1. Δέχεται μια κωδικοποιημένη διαδικασία <A,B> και επαναλαμβάνεται σε όλα τα χαρακτηριστικά που είναι ίσα ή μεγαλύτερα από το y(γραμμές 2-7). Εάν το χαρακτηριστικό δεν παρουσιάζεται στο B(γραμμή 3),υπολογίζει νέα διαδικασία C, D επεκτείνοντας το B με ένα γνώρισμα j(γραμμές 4,5). Αυτό αντιστοιχεί στο να πάρει μια τυπική έννοια της μορφής [4].

Ο αλγόριθμος 1 υπολογίζει νέες επίσημες διαδικασίες που προέρχονται από το <A,B>. Σημειώστε ότι ορισμένες από τις καινούριες διαδικασίες που λαμβάνονται με αυτόν τον τρόπο, μπορεί να είναι ίδιες. Γενικά μια απλή διαδικασία μπορεί να είναι το αποτέλεσμα του υπολογισμού της εξίσωσης [4] πολλαπλές φορές κατά την διάρκεια ολόκληρου του υπολογιστικού κύκλου. Για τον προσδιορισμό και την διαγραφή των επιπλέον διαδικασιών που έχουμε υπολογίσει, χρησιμοποιούμε ακριβώς το ίδιο τεστ κανονικοποίησης όπως το κανονικό CbO και την παράλληλη παραλλαγή PcbO. Στην περίπτωσή μας το τεστ κανονικοποίησης θα εμφανιστεί στην συνάρτηση ελαχιστοποίησης. Επίσης παρατηρούμε ότι η τιμή του B0 δεν χρησιμοποιείται από την συνάρτηση MAPCONCEPTS.

Η συνάρτηση REDUCECONCEPTS δέχεται μια κωδικοποιημένη πλειάδα

$$\textsc{ReduceConcepts}(\langle\langle B, j\rangle, \langle C, D\rangle\rangle) = \begin{cases} \langle\langle B, j+1\rangle, \langle C, D\rangle\rangle, & \text{if } B \cap Y_j = D \cap Y_j, \\ \textit{void-value}, & \text{otherwise.} \end{cases}$$

<<B,j>,<C,D>>και επιστρέφει μια τιμή σαν :

Επομένως, αν το τεστ κανονικοποίησης ικανοποιείται, τότε το ζευγάρι εισόδου <<B,j>,<C,D>> μειώνεται στο <<B,j+1>,<C,D>>,το οποίο θα χρησιμοποιηθεί στην επόμενη επανάληψη.

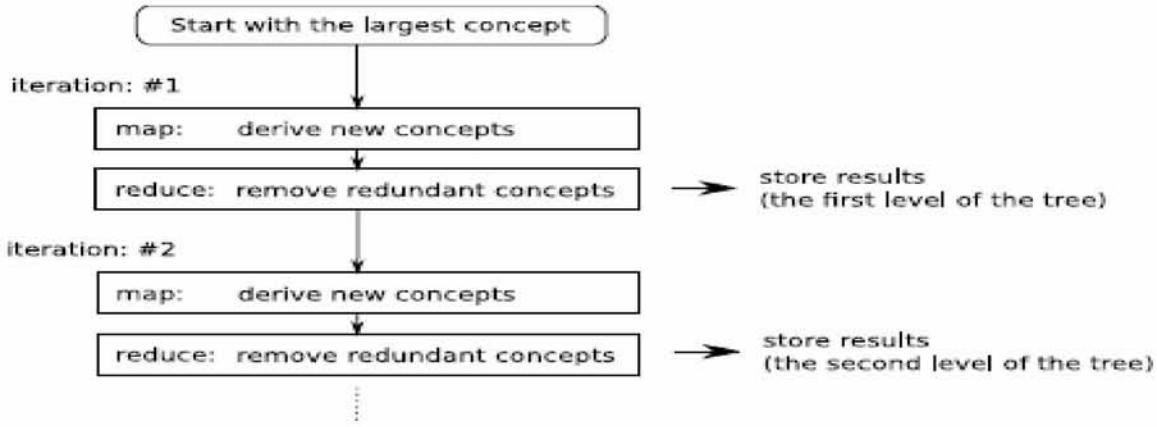
Algorithm 1: MAPCONCEPTS

Input: Pair $\langle key, value\rangle$ where key is $\langle B_0, y\rangle$ and $value$ is $\langle A, B\rangle$.

```
1 set result to ∅
2 for j = y to |Y| do
3     if j ∈ B then continue;
4     set C to A ∩ {j}↓
5     set D to C↑
6     set result to result ∪ {⟨⟨B, j⟩, ⟨C, D⟩⟩}
7 end
8 return result
```

Πηγή:P.Krajca,V.Vychodil,Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework
atbrox.com/about

Σημείωση 3:
Α) Κατά την διάρκεια των υπολογισμών, κάθε τιμή <B,j> του κλειδιού εμφανίζεται περισσότερες από μια φορές. Έτσι μπορούμε να θεωρήσουμε ότι η συνάρτηση REDUCECONCEPTS δέχεται μόνο ένα όρισμα εν αντιθέσει με ένα σύνολο από ορίσματα με το ίδιο κλειδί.
Β) Ειδικές υλοποιήσεις των πλαισίων mapreduce ,υποστηρίζονται για την διαχείριση των void-values.Στην πράξη, οι void-values δεν περιλαμβάνονται στα αποτελέσματα.
C) Η διαδικασία υπολογισμού τυπικών διαδικασιών μπορεί να θεωρηθεί σαν το χτίσιμο ενός δέντρου με τα εξής επίπεδα(εικόνα 3):



Εικόνα 3
Πηγή:P.Krajca,V.Vychodil,Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework
atbrox.com/about

6.5.Υλοποιήσεις και πειράματα

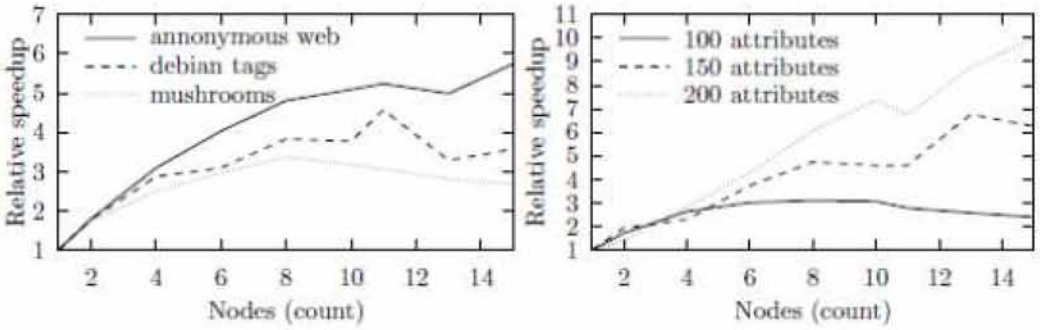
Έχουμε υλοποιήσει τον αλγόριθμο σαν μια εφαρμογή java χρησιμοποιώντας το πλαίσιο Hadoop Core[8] παρέχοντας υποδομή για map-reduce υπολογισμούς μαζί με κατανεμημένο σύστημα αρχείων για την αποθήκευση των δεδομένων εισόδου και των αποτελεσμάτων. Για τα πειράματά μας χρησιμοποιήσαμε cluster αποτελούμενο από 15 ανενεργούς επιτραπέζιους υπολογιστές εξοπλισμένους με Intel Core 2 Duo(3.0 GHz)επεξεργαστές, 2GB RAM,36 GB χώρο στον δίσκο και GNU/Linux. Παρουσιάζουμε δυο σύνολα προκαταρκτικών πειραμάτων, το πρώτο εστιάζει στον συνολικό χρόνο που χρειάστηκε για να υπολογίσει όλες τις διαδικασίες που υπάρχουν στον πραγματικό κόσμο συνόλου δεδομένων. Επιλέξαμε τρία σύνολα δεδομένων από το [9] και την αποθήκη μας με διάφορες ιδιότητες και μετρήσαμε τον χρόνο που πήρε για να υπολογίσει όλες τις τυπικές διαδικασίες χρησιμοποιώντας συστάδες που αποτελούνταν από 1,5 και 10 κόμβους. Τα αποτελέσματα φαίνονται στην παρακάτω εικόνα 4.

dataset	mushroom	debian tags	anon. web
size	8124 × 119	14315 × 475	32710 × 295
density	19 %	< 1 %	1 %
our (1 node)	1259	1379	2935
our (5 nodes)	436	426	830
our (10 nodes)	397	366	577
NextClosure	743	1793	10115

εικόνα 4 Χρόνος που χρειάστηκε για να υπολογιστούν όλες οι τυπικές έννοιες για διαφορετικά σύνολα δεδομένω(σε δευτερόλεπτα)
Πηγή: P.Krajca,V.Vychodil,Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework
atbrox.com/about

Για την σύγκριση συμπεριλάβαμε επίσης τον χρόνο τρεξίματος που παίρνει για να υπολογίσουμε όλες τις τυπικές διαδικασίες αν χρησιμοποιούσαμε τον αλγόριθμο Ganter's NextClosure[7]. Κατά την διάρκεια αξιολόγησης του κατανεμημένου αλγορίθμου, εκτός από την συνολική απόδοση, θα πρέπει να λάβουμε υπόψιν μας ένα σημαντικό χαρακτηριστικό που ονομάζεται κλιμάκωση. Με άλλα λόγια η ικανότητα να μειώσουμε τον χρόνο υπολογισμού χρησιμοποιώντας περισσότερους υπολογιστές. Στο δεύτερο σύνολο πειραμάτων εστιάζουμε σε αυτό το χαρακτηριστικό. Για να αναπαραστήσουμε την κλιμάκωση χρησιμοποιούμε σχετική επιτάχυνση, τον λόγο δηλαδή $S=T_1 /T_n$ όπου T_1 είναι ο χρόνος του υπολογισμού όταν τρέχουμε μόνο έναν υπολογιστή και T_n είναι ο χρόνος του υπολογισμού όταν λειτουργούν n υπολογιστές. Η θεωρητικά μέγιστη επιτάχυνση είναι ανάλογη με τον αριθμό των υπολογιστών. Η πραγματική επιτάχυνση είναι πάντα μικρότερη από την θεωρητική εξαιτίας πολλών παραγόντων συμπεριλαμβανομένου ειδικά της εναέριας επικοινωνίας ανάμεσα σε διαφορετικούς κόμβους,network throughput.

Η αριστερή γραφική παράσταση της εικόνα 5 μας δείχνει την κλιμάκωση του αλγορίθμου για επιλεγμένα σύνολα δεδομένων. Παρατηρούμε ότι για μικρούς αριθμούς κόμβων η επιτάχυνση είναι σχεδόν γραμμική, ενώ για αυξημένο αριθμό κόμβων η επιτάχυνση δεν είναι σημαντική. Στην περίπτωση των μεγάλων βάσεων δεδομένων(μανιτάρια-mushrooms)υπάρχει ακόμη μια μείωση της επιτάχυνσης, δηλαδή με αυξημένο αριθμό κόμβων η επιτάχυνση δεν αυξάνεται. Αυτό προξενείται από το overhead.Αυτό σημαίνει ότι το μέγεθος της συστάδας πρέπει να είναι ικανοποιητικό για το μέγεθος των δεδομένων εισόδου. Το γεγονός που φαίνεται δεξιά στο σχήμα 5 απεικονίζει την κλιμάκωση του αλγορίθμου σε τυχαία παραγόμενα περιβάλλοντα αποτελούμενα από 10000 αντικείμενα,100,150 και 200 χαρακτηριστικά, όπου η πυκνότητα των --1-- είναι 10%. Από την εικόνα 5(δεξί κομμάτι)έπεται ότι με αυξανόμενο μέγεθος δεδομένων, η κλιμάκωση αυξάνεται και για πίνακα δεδομένων μεγέθους 10000*200 ο υπολογισμός για συστάδα που αποτελείται από 15 μπορεί να λειτουργήσει ακόμη και 10x πιο γρήγορα από ότι με έναν μόνο υπολογιστή.



Εικόνα 5:Κλιμάκωση αλγορίθμου για επιλεγμένα σύνολα δεδομένων(αριστερά),κλιμάκωση αλγορίθμου σε τυχαία σύνολα δεδομένων(δεξιά)
Πηγή:P.Krajca,V.Vychodil,Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework
atbrox.com/about

6.6 Σχετικές Εργασίες-Συμπεράσματα

Αρκετοί παράλληλοι αλγόριθμοι για τον υπολογισμό τυπικών εννοιών έχουν προταθεί. Για παράδειγμα οι [5],[10] ή [12]. Γενικά, οι παράλληλοι αλγόριθμοι έχουν το μειονέκτημα ότι απαιτούν υλικό που περιέχει αρκετούς επεξεργαστές ή πυρήνες. Παρόλη την αλλαγή στην ανάπτυξη του υλικού προς την κατεύθυνση των πολυπύρηνων μικροεπεξεργαστών η δημιουργία υλικού με μεγάλες ποσότητες (περισσότεροι των 10) πυρήνων είναι ακόμη σχετικά ακριβή και σπάνια. Σε αντίθεση με το ότι οι κατανεμημένοι αλγόριθμοι μπορεί να τρέχουν σε ένα συνδυασμό υλικού. Παράλληλα προγράμματα συνήθως έχουν μια μικρότερη επιβάρυνση στον υπολογισμό σε σχέση με τα κατανεμημένα παρόλο που οι κατανεμημένοι αλγόριθμοι είναι πιο αποδοτικοί από πλευράς κόστους(μπορούν να εκτελεστούν από συνήθεις προσωπικούς υπολογιστές που έχουν πρόσβαση στο δίκτυο). Παρόλο που οι προαναφερόμενοι αλγόριθμοι μπορούν να τροποποιηθούν στις ad hoc κατανεμημένες εκδόσεις τους, από τα μέχρι στιγμής δεδομένα δεν υπάρχουν κατανεμημένες εφαρμογές αυτών των αλγορίθμων. Η προσέγγιση που ακολουθήσαμε μπορεί να θεωρηθεί ως μια απόδειξη της έννοιας του υπολογισμού τυπικών εννοιών από απομονωμένους κόμβους. Αποδείξαμε ότι ο αλγόριθμος είναι κλιμακωτός. Ως εκ τούτου υπάρχει η δυνατότητα εφαρμογής των τεχνικών της ανάλυσης των τυπικών εννοιών για πολύ μεγαλύτερα σύνολα δεδομένων από ότι στο παρελθόν.

Βιβλιογραφία

1. R. Belohlavek and V. Vychodil, “Discovery of optimal factors in binary data via a novel method of matrix decomposition,” Journal of Computer and System Sciences

(to appear).
2. A. Berry, J.-P Bordat, A. Sigayret, "A local approach to concept generation. Annals of Mathematics and Artificial Intelligence," 2007.
3. C. Carpineto and G. Romano, "Concept data analysis: Theory and applications," J. Wiley, 2004.
4. J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large cluster," Commun. ACM, 2008.
5. H. Fu and E.M. Nguifo, "A parallel algorithm to generate formal concepts for large Data," In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 394–401, Springer, Heidelberg (2004)
6. B. Ganter, "Two basic algorithms in concept analysis (Technical Report FB4-Preprint No. 831)," TH Darmstadt, 1984.
7. B. Ganter and R. Wille, "Formal concept analysis," Mathematical foundations, Springer, Berlin 1999.
8. Hadoop Core Framework: http://hadoop.apache.org/
9. S. Hettich and S.D. Bay, "The UCI KDD Archive University of California, Irvine, School of Information and Computer Sciences," 1999.
10. J.F.D. Kengue, P. Valtchev and C.T. Djamegni, "A parallel algorithm for lattice construction," In: Ganter, B., Godin, R. (eds.) ICFCA 2005. LNCS (LNAI), vol. 3403, pp. 249–264. Springer, Heidelberg 2005.
11. P. Miettinen, T. Mielik¨ainen, A. Gionis, G. Das and H. Mannila, "The discrete basis Problem," In: F¨urnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 335–346. Springer, Heidelberg, 2006.
12. P. Krajca, J. Outrata and V. Vychodil, "Parallel Recursive Algorithm for FCA," In: Belohlavek, R., Kuznetsov, S.O. (eds.) Proc. CLA 2008, vol. 433, pp. 71–82. CEUR WS, 2008, ISBN 978–80–244–2111–7.
13. P. Krajca, J. Outrata and V. Vychodil, "Parallel Algorithm for Computing Fixpoints of Galois Connections," Annals of Mathematics and Artificial Intelligence (submitted)
14. S. Kuznetsov, "Interpretation on graphs and complexity characteristics of a search for specific patterns," Automatic Documentation and Mathematical Linguistics 24(1), 37–45, 1989.
15. S. Kuznetsov, "A fast algorithm for computing all intersections of objects in a finite semi-lattice (Bystry_i algoritm postroeni_ vseh pereseqeni_i ob_ektov iz koneqno_i polurexetki, in Russian)," Automatic Documentation and Mathematical Linguistics 27(5), 11–21, 1993.
16. S.O. Kuznetsov, "Learning of simple conceptual graphs from positive and negative Examples," In: ˙Zytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 384–391. Springer, Heidelberg, 1999.
17. S. Kuznetsov and S. Obiedkov, "Comparing performance of algorithms for generating concept lattices," J. Exp. Theor. Artif. Int. 14, 189–216, 2002.
18. C. Lindig, "Fast concept analysis," In: Working with Conceptual Structures - Contributions to ICCS 2000, pp. 152–161, Shaker Verlag, Aachen, 2000.
P. Krajca and V.Vychodil, "Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework," Springer – Verlag Berlin, 2009.
Mapreduce & Hadoop Algorithms in Academic Papers (4th update – May 2011): www.atbrox.com/about

7 Μηχανή αναζήτησης ngram με πρότυπα που συνδυάζουν πληροφορία POS,Token,Chunk και NE

Οι Satosi Sekine και Kapil Dalwani αναπτύξανε ένα εργαλείο αναζήτησης ngrams που έχει εξαχθεί από μια πολύ μεγάλη συλλογή (το τωρινό σύστημα χρησιμοποίησε ολόκληρη την Wikipedia η οποία έχει 1.7 δισεκατομμύρια tokens). Το εργαλείο υποστηρίζει αναζητήσεις για ένα αυθαίρετο αριθμό wildcards(*) ή/και εξειδικευμένη αναζήτηση με ένα συνδυασμό token, POS, chunk(NP, VP, PP)και όνομα οντότητας(name entity)(NE). Στην έξοδο εμφανίζει τα ngrams που ταιριάζουν στην αναζήτηση, μαζί με τις συχνότητές τους καθώς επίσης και τις συναφείς εκφράσεις (π.χ. Προτάσεις, KWIC, λίστες και πληροφορίες για το id του εγγράφου). Η διαδικασία αυτή χρειάζεται ένα κλάσμα του δευτερολέπτου για μια αναζήτηση σε ένα μονοπύρηνο περιβάλλον Linux(1GBμνήμη, 500 Gb δίσκο).

Αυτό το σύστημα είναι μια επέκταση παλαιότερης έκδοσης μιας μηχανής αναζήτησης n-gram (Sekine 08). Το προηγούμενο σύστημα μπορούσε να χειριστεί μόνο tokens και απεριόριστα wildcards σε μια αναζήτηση όπως "* ιδρύθηκε στο *". Ωστόσο, εφόσον μπορούμε να περιορίσουμε τα wildcards με την βοήθεια των POS, chunk ή NE είναι αρκετά χρήσιμο το φιλτράρισμα του "θορύβου". Για παράδειγμα, το νέο σύστημα μπορεί να κάνει αναζήτηση για "NE=COMPANY ιδρύθηκε το POS=CD". Αυτή η λεπτομερής εξειδίκευση μειώνει τον αριθμό των αποτελεσμάτων σε λιγότερα από το μισό και αποφεύγει τα ngrams τα οποία έχουν κόμμα ή ένα κοινό ουσιαστικό στην πρώτη θέση ή πληροφορίες τοποθεσίας στην τελευταία θέση.

Το νέο σύστημα μπορεί να προσφέρει πληροφορίες στα έγγραφα από τα οποία τα ταιριασμένα ngrams έχουν εξαχθεί. Στο τωρινό σύστημα το οποίο χρησιμοποιεί το Wikipedia, η πληροφορία του εγγράφου είναι ο τίτλος της Wikipedia σελίδας. Για παράδειγμα: μπορούμε να βρούμε το όνομα του ατόμου στον τίτλο του ngram που ταιριάζει για την αναζήτηση "Γεννήθηκε το *".Επίσης είναι χρήσιμο να έχουμε έναν pointer επιστροφής σε όλο το άρθρο που περιέχει τα ταιριασμένα ngrams.

Η δομή του ευρετηρίου έχει αλλάξει εντελώς από την δομή trie του προηγούμενου συστήματος σε μια ανεστραμμένη δομή ευρετηρίου σε συνδυασμό με έναν επιπρόσθετο μηχανισμό ελέγχου. Το μεγέθους του ευρετηρίου έχει μειωθεί σημαντικά από 2.4 TB σε 500GB με μια μικρή θυσία στην ταχύτητα της αναζήτησης.

7.1 Υπόβαθρο

Ανακάλυψη Μεγάλης κλίμακας γλωσσολογικής γνώσης χρειάζεται για να στηρίξει τη σημασιολογική ανάλυση για τις εφαρμογές NLP.Αυτό είναι το επονομαζόμενο "knowledge bottleneck"πρόβλημα και πολλοί ερευνητές έχουν προσπαθήσει να το λύσουν χρησιμοποιώντας στατιστικές μεθόδους σε μια μεγάλη συλλογή δεδομένων(Hearst 92,Collins and Singer 99,Brin 99,Haegawa και άλλοι 04).Για παράδειγμα το λεξικο-συντακτικό πρότυπο "NP such as NP" μπορεί να εξάγει υπόνυμες(hyponym) σχέσεις(πχ η πατάτα είναι υπόνυμη του λαχανικού) και γενικότερα πλαίσια ανάμεσα σε δυο ονόματα οντοτήτων(Hasegawa και άλλοι 04).Ένα από τα

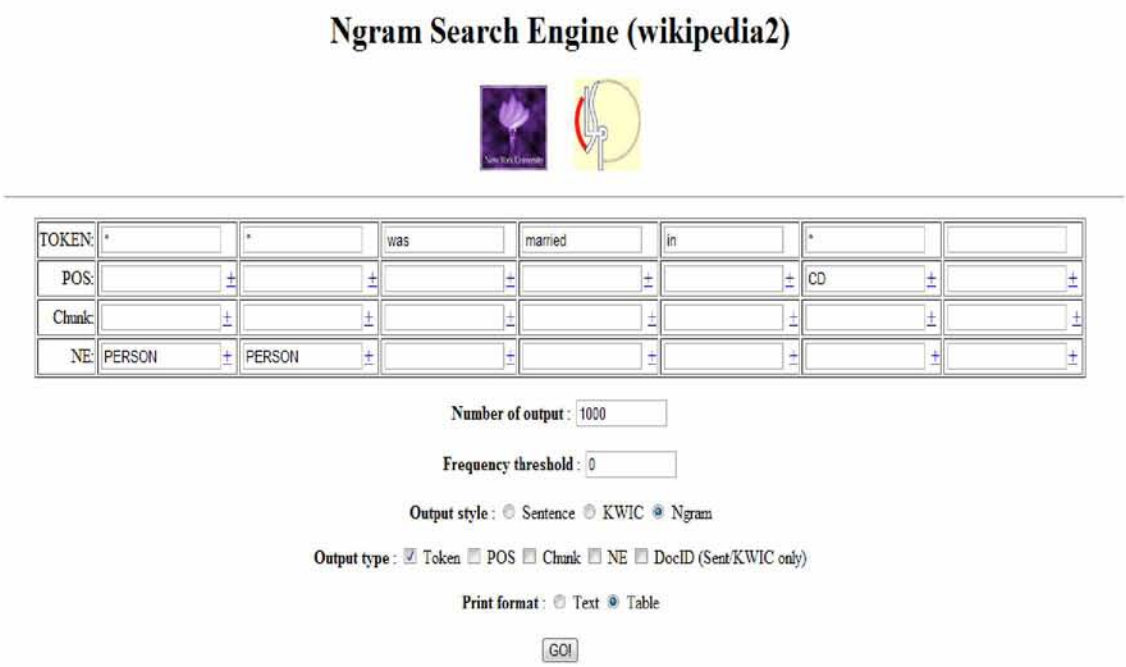
μεγαλύτερα προβλήματα είναι η αναζήτηση. Χρειαζόμαστε ικανότητα για να αναζητήσουμε πρότυπα(patterns) σε μια μεγάλη συλλογή που είναι γρήγορη και όσο το δυνατόν πιο ευέλικτη. Μια από τις λύσεις είναι να χωρίσουμε ολόκληρη τη συλλογή σε μικρά κομμάτια και να αναθέσουμε μια CPU σε κάθε τμήμα για να ψάξει το πρότυπο και να συγκεντρώσει τα αποτελέσματα σε ένα mapreduce παράδειγμα. Ωστόσο αυτή η προσέγγιση απαιτεί μεγάλο αριθμό μηχανών ο οποίος δεν είναι προσιτός(οικονομικά) για πολλούς ακαδημαϊκούς ερευνητές.

Μια άλλη λύση στο πρόβλημα είναι να χρησιμοποιήσουμε διαθέσιμα προγράμματα αναζήτησης, όπως το Lucene. Ωστόσο η ευελιξία είναι απαραίτητη, για παράδειγμα χρειαζόμαστε την ικανότητα να ταιριάζουμε ακριβώς μια ngram αναζήτηση με τα wildcards και την ικανότητα να παρέχουμε επιπρόσθετες πληροφορίες όπως POS, chunk ή/και NE. Βρήκαμε ότι είναι εφικτό να τροποποιήσουμε το Lucene για να πετύχουμε τον σκοπό μας αλλά καταλήξαμε ότι ήταν απαραίτητο να αναπτύξουμε το δικό μας σύστημα αναζήτησης έτσι ώστε να μπορούμε να το επεκτείνουμε στο μέλλον. Το σύστημα του Resnik (03)έχει παρόμοιες λειτουργικότητες με εκείνες που παρέχονται σε αυτό το σύστημα αλλά πιστεύουμε ότι έχουμε βελτιωθεί σε αυτό όσον αφορά την επεκτασιμότητα και την ταχύτητα. Το demo(δοκιμαστικό) και η ιστοσελίδα του project δεν είναι πια διαθέσιμα.

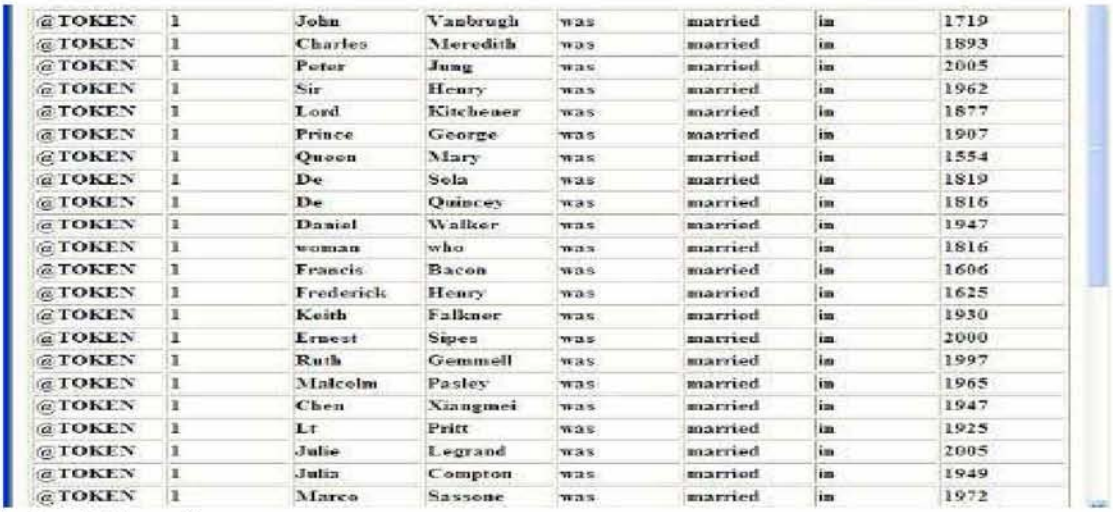
7.2.Στιγμυότυπο

Οι εικόνες 1, 2, 3 δείχνουν στιγμιότυπα του συστήματος. Η εικόνα 1 είναι η σελίδα αναζήτησης, η εικόνα 2 είναι η σελίδα αποτελεσμάτων αναζήτησης των ngrams και η εικόνα 3 είναι η σελίδα των αποτελεσμάτων KWIC με έγγραφα ID(Π.χ. τίτλος καταχώρησης Wikipedia).

Στην σελίδα αναζήτησης στην εικόνα 1 ο χρήστης πληκτρολογεί την αναζήτηση ngram με tokens, POS, chunk ή NE πληροφορία μέχρι 7 gram.Ο χρήστης μπορεί επίσης να προσδιορίσει τον αριθμό των αποτελεσμάτων, το κατώφλι συχνότητας (να εμφανίζεται η κατώτατη συχνότητα), το στυλ αποτελεσμάτων (πρόταση, KWIC ή ngram), ο τύπος των αποτελεσμάτων (token, POS, chunk, NE ή/και πληροφορίες εγγράφων σε περίπτωση πρότασης ή αποτελέσματος KWIC) και η μορφή εκτύπωσης (σε κείμενο ή πίνακα). Τα αποτελέσματα θα εμφανίζονται σύμφωνα με τις προδιαγραφές.



Εικόνα 1:Στιγμιότυπο
Πηγή:**http://linserv1.cims.nyu.edu:23232/ngram_wikipedia2K/**



@TOKEN	1	John	Vanbrugh	was	married	in	1719
@TOKEN	1	Charles	Meredith	was	married	in	1893
@TOKEN	1	Peter	Jung	was	married	in	2005
@TOKEN	1	Sir	Henry	was	married	in	1962
@TOKEN	1	Lord	Kitchener	was	married	in	1877
@TOKEN	1	Prince	George	was	married	in	1907
@TOKEN	1	Queen	Mary	was	married	in	1554
@TOKEN	1	De	Sola	was	married	in	1819
@TOKEN	1	De	Quincey	was	married	in	1816
@TOKEN	1	Daniel	Walker	was	married	in	1947
@TOKEN	1	woman	who	was	married	in	1816
@TOKEN	1	Francis	Bacon	was	married	in	1606
@TOKEN	1	Frederick	Henry	was	married	in	1625
@TOKEN	1	Keith	Falkner	was	married	in	1930
@TOKEN	1	Ernest	Sipes	was	married	in	2000
@TOKEN	1	Ruth	Gemmell	was	married	in	1997
@TOKEN	1	Malcolm	Pasley	was	married	in	1965
@TOKEN	1	Chen	Xiangmei	was	married	in	1947
@TOKEN	1	Lt	Pritt	was	married	in	1925
@TOKEN	1	Julie	Legrand	was	married	in	2005
@TOKEN	1	Julia	Compton	was	married	in	1949
@TOKEN	1	Marco	Sassone	was	married	in	1972

Εικόνα 2:Στιγμιότυπο
http://linserv1.cims.nyu.edu:23232/cgi-bin/ngram_wikipedia2/wikipedia2.cgi?key1t=*&key2t=*&key3t=was&key4t=married&key5t=in&key6t=*&key7t=&key1p=&key2p=&key3p=&key4p=&key5p=&key6p=CD&key7p=&key1c=&key2c=&key3c=&key4c=&key5c=&key6c=&key7c=&key1n=PERSON&key2n=PERSON&key3n=&key4n=&key5n=&key6n=&key7n=&print_max=1000&freq_threshold=0&print_style=3&print_type_token=1&print_format=table&key=

Abdul Aziz al-Hakim	assassinated in August 2003 in Najaf . #BOS# Background #BOS#	He was born in 1950	, the son of Grand Ayatollah Muhsin Al - Hakim
George Borowski	who may have been a reference to Borowski . #BOS#	He was born in 1950	, in Wrexham in North Wales to Russian and Polish
Dinu Patriciu	member of the National Liberal Party of Romania . #BOS#	He was born in 1950	, is married and has two daughters : Ana and
Lou Jiwei	Biography #BOS# Lou is a native of Beijing . #BOS#	He was born in 1950	, joined the Communist Party of China in 1973 ,
Andrew Ridgway	14 June 2006 after a long military career . #BOS#	He was born in 1950	, educated at Hele ' s School , Exeter ,
Steve Houben	Houben is a Belgian jazz saxophonist and flutist . #BOS#	He was born in 1950	. #BOS# In the mid 1970s , he attended the
Oleg Betin	. #BOS# He is a member of NDR . #BOS#	He was born in 1950	. #BOS# In 1995 he became governor . #BOS# He
Charlie Kosei	the soundtracks of the Lupin III anime series . #BOS#	He was born in 1950	. #BOS# He also contributed his voice to the English
David Krutz	of several books regarding the American Civil War . #BOS#	He was born in 1950	and was raised in Little Falls , New York .
Frank McDonald	Environment Editor of The Irish Times . #BOS# Life #BOS#	He was born in 1950	and educated at Kelly ' s Private School , Cabra
Patric Laurence Dickinson	' s Richmond Herald of Arms in Ordinary . #BOS#	He was born in 1950	and educated at Marling School in Stroud , Gloucestershire before
Bishnu Pada Ray	elected to 13th Lok Sabha from Andaman and Nicobar Islands	He was born in 1950	in 24 Parganas in West Bengal . #BOS# External links
David Eggby	Eggby #BOS# David Eggby is a British cinematographer . #BOS#	He was born in 1950	in London . #BOS# He received the Cinematographer of the
David Unger	a famous Guatemalan - American author and translator . #BOS#	He was born in 1950	in Guatemala City . #BOS# In 1955 , he emigrated

Medieval churches of York	the small tower of its predecessor (in which Sir	John Vanbrugh was married in 1719	) . #BOS# This dates back to the twelfth century
Charles Meredith	forced to retire from business life in 1924 . #BOS#	Charles Meredith was married in 1893	to Elspeth Hudson (1858 - 1936) , daughter
Peter Chung	most prominent forums include ILX and Monican Spies . #BOS#	Peter Jung was married in 2005	. #BOS# He should not be confused with Peter Chun
Henry Milton Taylor	on 1 January 1992 . #BOS# Marriage and children #BOS#	Sir Henry was married in 1962	to the former Elua Mae Sisco of Arcadia , Florida
Henry Kitchener, 2nd Earl Kitchener	of Flax - producing estates in the colony . #BOS#	Lord Kitchener was married in 1877	to Eleanor Fanny Lushington (died 1897) with whom
Prince George of Greece and Denmark	Assembly declared " enosis " with Greece unilaterally . #BOS#	Prince George was married in 1907	to Princess Marie Bonaparte , daughter of Prince Roland Bonaparte
Shilling (United Kingdom)	#BOS# TUN #BOS# No shillings were struck in England until	Queen Mary was married in 1554	, however Irish shillings with Mary ' s portrait were
David de Aaron de Sola	published in 1855 and afterward translated into English . #BOS#	De Sola was married in 1819	to Rica Meldola , the eldest daughter of Haham Raphael
Thomas de Quincey	and which is now a popular tourist attraction . #BOS#	De Quincey was married in 1816	, and soon after , having no money left ,
Daniel Walker	but his request was not granted . #BOS# Family #BOS#	Daniel Walker was married in 1947	to Roberta Dowse , a Catholic school teacher from Kenosha

Εικόνα 3:Στιγμιότυπο τρεξίματος

7.3.Δεδομένα και αλγόριθμοι αναζήτησης.

Χρησιμοποιήσαμε το wikipedia ως στόχο για την αναζήτησή μας σε αυτό το σύστημα. Το google ngram θα μπορούσε επίσης να χρησιμοποιηθεί, αλλά επειδή δεν περιέχει τις αυθεντικές προτάσεις χρησιμοποιήσαμε το wikipedeia και δημιουργήσαμε μόνοι μας δεδομένα ngrams από αυτό, για να δείξουμε τα αρχικά δεδομένα από όπου τα ngrams εξάχθηκαν. Είναι στατικά html έγγραφα από το Wikipedia όπως από το 18.12 ,june,8,2008 έκδοση που παρέχεται από το ακόλουθο url http://static.wikipedia.org/downloads/2008-06/en/.Έχει 1,7 δισεκατομμύρια tokens ,200

εκατομμύρια προτάσεις και 2,4 εκατομμύρια άρθρα. Οι προτάσεις επισημάνθηκαν από τον Stanford POS tagger και τον NE tagger(Stanford tagger) και τα chunks που έχουν ανατεθεί από το OAK σύστημα. Τα ίδια δεδομένα(στην πραγματικότητα τα τρέχοντα διαθέσιμα δεδομένα στο site έχουν περισσότερα σχόλια από τα δεδομένα που εξηγήθηκαν εδώ)και είναι διαθέσιμα στο ακόλουθο url http://nlp.cs.nyu.edu/wikipedia-data/. Οι αριθμοί από τα διακριτά ngrams φαίνονται στο πίνακα 1. Οι αριθμοί είναι συγκρινόμενοι με τα δεδομένα του google ngram καθώς δεν έχουμε κατώφλι συχνότητας. Φτιάξαμε 7grams αντί για 5grams του google ngram.

Αριθμός των διακριτών ngrams

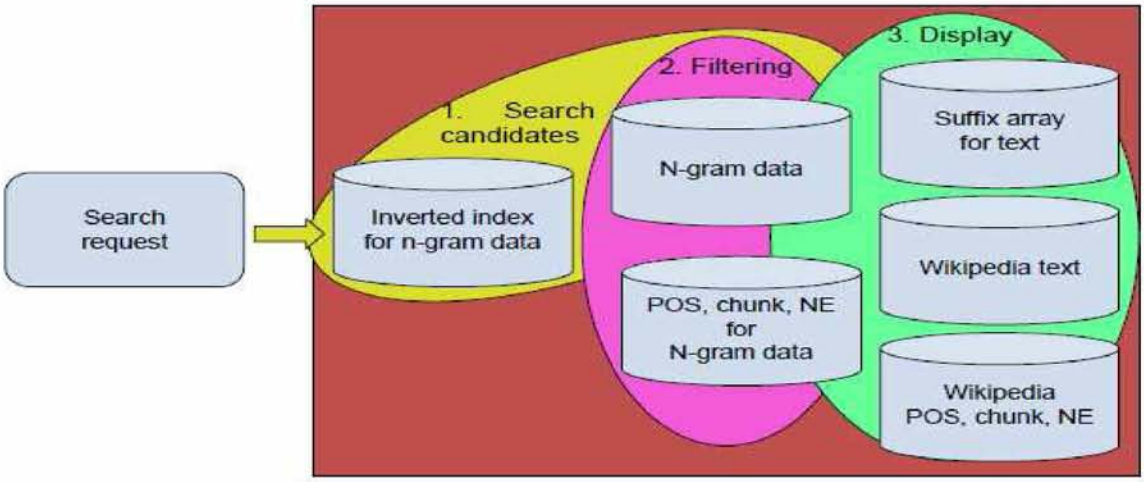
N	Wikipedia ngrams	Google ngram
1	8M	13M
2	93M	315M
3	377M	977M
4	733M	1,314M
5	1,006M	1.176M
6	1,173M	-
7	1,266M	-
#	0	30

#κατώφλι συχνότητας

Πηγή:S.Sekine,K.Dalwani,Ngram Search Engine with Patterns Combining Token,POS,Chunk and NE Information

7.4 Επισκόπηση αλγορίθμου

Η εικόνα 4 δείχνει μια επισκόπηση των βημάτων της μηχανής αναζήτησης ngram και των δεδομένων. Βασικά υπάρχουν τρία βήματα στην αναζήτηση: 1) αναζήτηση υποψηφίων ngrams όπου το σύστημα αναζητά υποψήφια ngrams που ταιριάζει με τα tokens των αναζητήσεων 2) φιλτράρισμα όπου τα υποψήφια ngrams φιλτράρονται με την βοήθεια επιπρόσθετων πληροφοριών(POS, chunk, NE) και 3) εμφάνιση των αποτελεσμάτων στον χρήστη. Στις ακόλουθες υποενότητες θα περιγράψουμε κάθε βήμα και δεδομένα με λεπτομέρειες.



Εικόνα 4 Επισκόπιση αλγορίθμου
http://nlp.cs.nyu.edu/wikipedia-data/

7.5 Αναζήτηση υποψηφίου ngram

Η εύρεση υποψηφίων ngrams είναι το πρώτο βήμα της αναζήτησης. Προσπαθεί να φτιάξει μια λίστα από ngrams χρησιμοποιώντας ένα ανεστραμμένο ευρετήριο για τα tokens που δίνονται στην αναζήτηση. Εξαιτίας αυτού, η αναζήτηση χρειάζεται να έχει τουλάχιστον ένα token. Το ανεστραμμένο ευρετήριο είναι μια στάνταρ τεχνική για αναζήτηση στοιχείων(items). Στην περίπτωσή μας, το ανεστραμμένο ευρετήριο δημιουργήθηκε για tokens που βρίσκονται σε κάθε θέση σε όλο το μήκος των ngrams. Το ανεστραμμένο ευρετήριο, βασικά περιέχει ένα σετ από ngramID που έχουν ένα συγκεκριμένο token σε μια συγκεκριμένη θέση ενός συγκεκριμένου μήκους ngram. Επειδή ο αριθμός των ngrams για κάθε ευρετήριο ποικίλει (από 55 εκατ που είναι ο αριθμός των "," σε μια συγκεκριμένη θέση μέχρι των αριθμό των πολύ σπανίων tokens όπως "Mizuk" ή "consiety". Στην προσπάθεια να γλιτώσουμε χρόνο και χώρο στον δίσκο χρησιμοποιήσαμε 3 τύπους λίστας υλοποίησης (ανεστραμμένο ευρετήριο για τα token), bitmap, λίστα από ngramIDs και κωδικοποίηση σε δείκτη.

Η τεχνική bitmap χρησιμοποιείται μόνο σε αυτά τα ngrams των οποίων η συχνότητα είναι πάνω από 1%.Για παράδειγμα ο αριθμός των διακριτών 7grams είναι 1,27 δισεκατομμύρια tokens των οποίων οι συχνότητες είναι πάνω από 12,7 εκατομμύρια. Εξαιτίας του περιορισμού στον χρόνο υλοποίησης, δεν χρησιμοποιήσαμε καμία τεχνική συμπίεσης. Χρησιμοποιήσαμε 1,27 δισεκατομμύρια bits στην περίπτωση των 7grams.

Η αναζήτηση των πληροφοριών είναι πολύ γρήγορη παρόλο το μέγεθος του ανεστραμμένου ευρετηρίου για το token. Αν η συχνότητα του token είναι ένα, αποθηκεύουμε την πληροφορία στην περιοχή του δείκτη ρυθμίζοντας το κορυφαίο bit(σημειώστε ότι ο μέγιστος αριθμός ngram που είναι 1,27 δισεκατομμύρια μπορούν να εκφραστούν με 31 bits). Αλλιώς χρησιμοποιούμε λίστα. Επειδή μπορεί να δημιουργηθεί

εκ των προτέρων, μπορούμε να χρησιμοποιήσουμε στατική λίστα αντί για δυναμική η οποία χρειάζεται περισσότερο χώρο. Όταν υπάρχουν περισσότερα από δυο tokens στην αναζήτηση, πρέπει να ταιριάξουμε τις λίστες. Ταιριάζοντας δυο λίστες μήκους n και m μπορεί να υλοποιηθεί σε min(O(n+m), O(nlog(m), O(mlog(n))).Αν τα n ή m δεν είναι πολύ μικρά πρέπει να χρησιμοποιήσουμε τον αλγόριθμο που ταιριάζει κοιτώντας την λίστα σειριακά στον χρόνο O(n+m),το οποίο δεν είναι πολύ γρήγορο παρόλο που μπορούμε να ταξινομήσουμε το ευρετήριο από πριν. Για να επιταχύνουμε το ταίριασμα του ευρετηρίου εφαρμόσαμε τον αλγόριθμο "look ahead"(Moffat ,Zbel 96). Όταν βρούμε ένα ταίριασμα, προσπερνάμε κάποια από τα στοιχεία στην λίστα και μετακινούμαστε στο στοιχείο σε μια συγκεκριμένη απόσταση. Αν το νέο στοιχείο που βρέθηκε είναι ακόμα μικρότερο από το στοιχείο που ψάχνουμε, μπορούμε να κερδίσουμε χρόνο από το να κοιτάμε στα στοιχεία που έχουν προσπεραστεί. Αν το στοιχείο που έχουμε μετατοπιστεί είναι μεγαλύτερο από αυτό που ψάχνουμε θα πάμε πίσω στην αρχική θέση και η αναζήτηση συνεχίζει από το επόμενο στοιχείο. Σπαταλάμε μόνο ένα ‘’κοίταγμα’’. Βασιζόμενοι στον αλγόριθμο είναι πιο αποτελεσματικό να κοιτάξουμε την τετραγωνική ρίζα του μεγέθους του ευρετηρίου όταν προχωράς τον δείκτη στο ταίριασμα του ευρετηρίου.

7.6 Φιλτράρισμα

Το δεύτερο βήμα(φιλτράρισμα) χρειάζεται να ταιριάξει το ngram με τη ζητούμενη πληροφορία POS, chunk και NE. Αν εφαρμόσουμε όλες αυτές τις πληροφορίες για κάθε ngram χρησιμοποιώντας ένα byte το καθένα, χρειαζόμαστε 21 επιπλέον bytes στην περίπτωση του 7gram. Σαν αποτέλεσμα, χρειαζόμαστε 27 GB. Έτσι, κωδικοποιήσαμε τις πληροφορίες με βάση τα ξεχωριστά tokens του κάθε ngram. Για παράδειγμα για 7grams οι πληροφορίες POS για τα 7 tokens καταλαμβάνουν 7bytes και γιατί για κάθε εγγραφή πληροφορίας POS για ένα μοναδικό token χρειαζόμαστε ένα byte. Ωστόσο ο πραγματικός αριθμός των συνδυασμών των 7 pos δεν είναι τόσο μεγάλος επειδή υπάρχουν πολλά ngrams που έχουν τα ίδια POS πρότυπα. Για παράδειγμα ο αριθμός των προτύπων POS για 7gram είναι 125 εκατομμύρια και ο αριθμός των προτύπων POS ΓΙΑ 3gram είναι 61 χιλιάδες, τα οποία μπορούν να κωδικοποιηθούν σε λιγότερο από 4 bytes προκαλώντας μεγάλη μείωση στο μέγεθος του δίσκου(200 MB σύνολο).

Ελέγχουμε αν το υποψήφιο ngram μπορεί να ταιριάξει με την αναζήτηση, βρίσκοντας αν οι πληροφορίες POS, chunk ή NE ικανοποιούν την αναζήτηση για τις ζητούμενες πληροφορίες.

7.7 Παρουσίαση αποτελεσμάτων

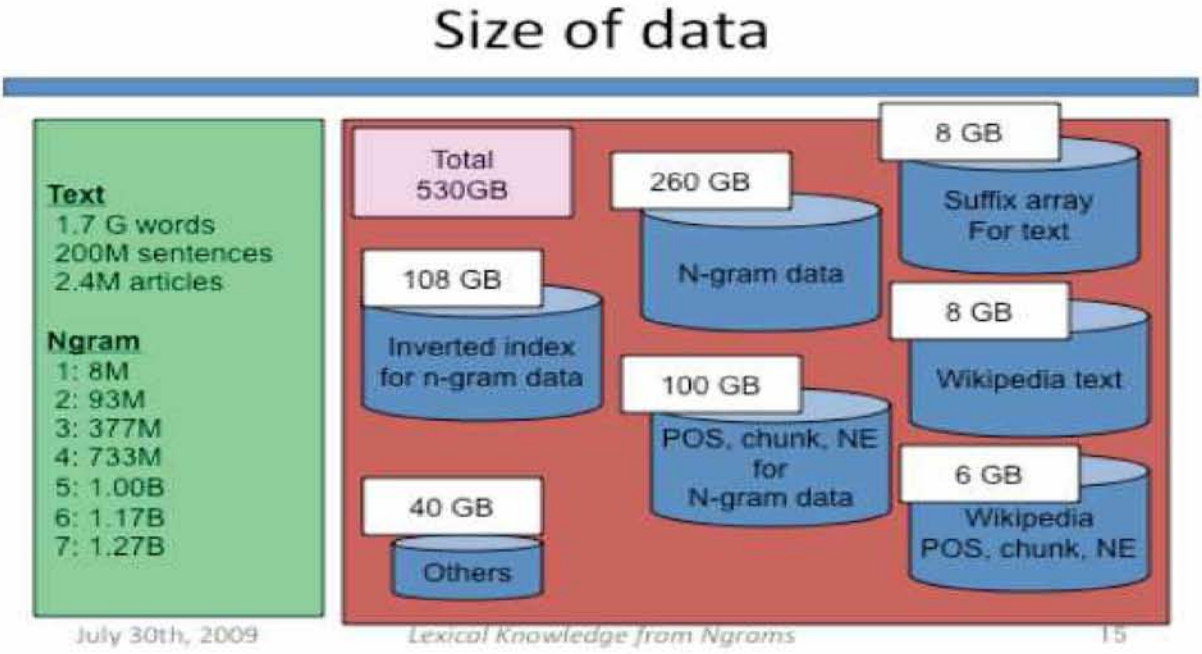
Το τρίτο βήμα(παρουσίαση αποτελεσμάτων) πραγματοποιείται όταν τα ngrams που πρέπει να παρουσιαστούν έχουν βρεθεί. Ο χρήστης μπορεί επιπρόσθετα να προσδιορίσει τις ακόλουθες επιλογές στην αναζήτηση ngram:

- Αριθμός αποτελεσμάτων(προεπιλογή 1000).
- Κατώφλι συχνότητας(προεπιλογή 3)
- Στυλ αποτελεσμάτων : πρόταση, KWIC ή Ngram
- Τύπος αποτελεσμάτων : token, POS, chunk, NE, DocID
- DocID μπορεί να δουλέψει μόνο για πρόταση και KWIC.
- Μορφή εκτύπωσης : κείμενο, πίνακας, html table

Η παρουσίαση θα γίνει σύμφωνα με αυτές τις επιλογές. Τα Ngrams ταξινομούνται εκ των προτέρων με βάση την συχνότητα έτσι ώστε να μπορούν να παρουσιαστούν εύκολα με σειρά συχνότητας. Ο πίνακας που εμφανίζεται χρησιμοποιείται για να παρουσιάσει γρήγορα την πρόταση και την λίστα KWIC.Κάθε ngram έχει πληροφορίες της αρχικής και τελικής θέσης, του τελικού πίνακα με τον οποίο ταιριάζει έτσι ώστε να μπορεί να βρει τις ταιριαστές προτάσεις πολύ γρήγορα. Η πληροφορία POS,chunk και NE αποθηκεύεται παράλληλα με την πληροφορία κειμένου έτσι ώστε να εμφανιστεί γρήγορα όταν ζητηθεί. Οι πληροφορίες των εγγράφων συμπεριλαμβανομένου του τίτλου του εγγράφου (στην περίπτωση μας ο τίτλος της σελίδας Wikipedia) εμφανίζονται χρησιμοποιώντας την πληροφορία που είναι αποθηκευμένη στο κείμενο.

7.8 Μέγεθος δεδομένων

Χρησιμοποιούμε 6 είδη δεδομένων όπως φαίνεται στην εικόνα 6. Στην αναζήτηση χρησιμοποιείται ανεστραμμένο ευρετήριο και το μέγεθος είναι 108 GB. Περιέχει bitmap ευρετήριο και ευρετήριο λίστας. Τα δεδομένα ngram συμπεριλαμβανομένων των πληροφοριών token και την θέση στο τελικό πίνακα που εμφανίζεται καταλαμβάνουν 260GB. Για μετατροπή από unigram σε 7gram που χρησιμοποιείται στο φιλτράρισμα, χρησιμοποιείται πληροφορία POS,chunk και NE που καταλαμβάνει 100 GB. Όπως εξηγείται στο section 4.4, οι πληροφορίες είναι συμπιεσμένες. Το μέγεθος του τελικού πίνακα που εμφανίζεται και των δεδομένων του κειμένου είναι 8gb το καθένα (καθώς κάθε token και δείκτης είναι κωδικοποιημένα σε 4 bytes και ολόκληρο το κείμενο είναι 1,7 δισεκατομμύρια) και το μέγεθος του POS,chunk και NE είναι 2 GB το καθένα(καθώς κάθε πληροφορία κωδικοποιείται σε 1 byte). Το μέγεθος άλλων πληροφοριών, όπως πληροφορίες εγγράφων, πληροφορίες λεξικών, POS, chunk και NE κτλ είναι 40 Gb. Έτσι, το συνολικό μέγεθος των δεδομένων του συστήματος είναι περίπου 500 GB. Καταλαβαίνουμε ότι το μέγεθος μπορεί να μειωθεί εύκολα χωρίς μεγάλη απώλεια στην ταχύτητα και στην χρηστικότητα.



Εικόνα 6 μέγεθος δεδομένων.
S.Sekine,K.Dalwani,Ngram Search Engine with Patterns Combining Token,POS,Chunk and NE Information

7.9 Αξιολόγηση και demo συστήματος

Για να αξιολογήσουμε την ακρίβεια και την ταχύτητα του συστήματος, δημιουργήσαμε 600 δείγματα αναζητήσεων και δοκιμάσαμε το σύστημα. Τα δείγματα εξήχθησαν από υπάρχοντα ngrams αντικαθιστώντας το 0 μέχρι 2 tokens με μπαλαντέρ, POS, chunk ή NE τυχαία και βλέποντας αν το αρχικό ngram συμπεριλαμβάνεται στα αποτελέσματα αναζήτησης. Να σημειώσουμε ότι είναι πιθανό να συμπεριλάβουμε επιπλέον ngrams εξαιτίας της γενίκευσης, αλλά αυτό δεν θα έπρεπε να κρίνεται σαν λάθος. Το σύστημα τρέχει χωρίς κανένα λάθος αποτέλεσμα και ο μέσος χρόνος εκτέλεσης για κάθε ερώτηση είναι 0,34 δευτερόλεπτα.
Demo url:http://nlp.cs.nyu.edu/nsearch

Βιβλιογραφία

M. Collins and Y. Singer, "Unsupervised Models for Named Entity Classification," , In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-99), 1998.

M. A. Hearst, "Automatic Acquisition of Hyponyms from Large Text Corpora," In Proceedings of Conference on Computational Linguistics (COLING-92), Nantes, France, 1992.

T. Hasegawa, S. Sekine, R. Grishman, "Discovering Relations among Named Entities from Large Corpora," In the proceedings of Association of Computational Linguistics (ACL-04). Barcelona, Spain, 2004.

A. Moffat and J. Zobel, "Inverted Files for Text Search Engines," ACM Computing Surveys, 38(2):1-56, July 2006.

P. Resnik and A. Elkiss, "The Linguist's Search Engine: Getting Started Guide," Technical Report: LAMP-TR-108/CS-TR-4541/UMIACS-TR-2003-109, University of Maryland, College Park, 2003.

S. Sekine, "A Linguistic Knowledge Discovery Tool," In Proceeding of COLING08.
Stanford tagger: http://nlp.stanford.edu/software/tagger.shtml, 2008

OAK system: http://nlp.cs.nyu.edu/oak
Wikipedia tagged data: http://nlp.cs.nyu.edu/wikipedia-data

S.Sekine and K.Dalwani, "Ngram Search Engine with Patterns Combining Token,POS,Chunk and NE Information

8. Ανάλυση Κλιμακωτών RDF γράφων στο MapReduce

Για να εκμεταλλευτούμε τον αυξανόμενο αριθμό των δεδομένων RDF στην λήψη αποφάσεων, υπάρχει αυξανόμενη ανάγκη για αναλυτικό στυλ επεξεργασίας τέτοιου είδους δεδομένων. Τα δεδομένα RDF μοντελοποιούνται σαν ένα γράφημα που αντιπροσωπεύει μια συλλογή δυαδικών σχέσεων(τριπλάσια). Σε αυτό το γενικότερο πλαίσιο, αναλυτικές αναζητήσεις μπορούν να ερμηνευτούν σαν να αποτελούνται από τρεις κυρίως κατασκευές ,δηλαδή ταίριασμα προτύπου, ομαδοποίηση και άθροισμα, και απαιτούν διάφορες λειτουργίες για να τις επανασυναρμολογήσουμε σε n-δικές σχέσεις σχετικές με την δεδομένη αναζήτηση σε αντίθεση με τα παραδοσιακά συστήματα OLAP, όπου τα δεδομένα οργανώνονται κατάλληλα. MapReduce συστήματα παράλληλης επεξεργασίας, όπως το Pig έχουν αποκτήσει μεγάλη επιτυχία στο να επεξεργάζονται κλιμακωτές αναλυτικές εργασίες. Αλλά αυτά τα συστήματα προσφέρουν μόνο αλγεβρικούς τελεστές οι οποίοι θα απαιτούσαν μια επαναληπτική διαδικασία συναρμολόγησης με n πλειάδες στην οποία ενδιάμεσα αποτελέσματα πρέπει να υλοποιηθούν. Αυτό οδηγεί σε υψηλά κόστη I/O και επηρεάζει αρνητικά την απόδοση. Προτείνουμε UDFs τα οποία πρώτον παραγοντοποιεί αναλυτική επεξεργασία σε γράφους RDF με έναν τρόπο που επιτρέπει πιο παραλληλοποιημένης επεξεργασίας. Δεύτερον εκτελούν μια επεξεργασία πρόβλεψης για να μειώσουμε το κόστος μεταγενέστερων τελεστών στο πλάνο εκτέλεσης της αναζήτησης. Αυτές οι λειτουργίες έχουν ολοκληρωθεί στην βιβλιοθήκη συναρτήσεων Pig Latin και τα πειραματικά αποτελέσματα δείχνουν πάνω από 50% βελτίωση στον χρόνο εκτέλεσης για συγκεκριμένες τάξεις αναζητήσεων. Μια σημαντική επίδραση αυτής της εργασίας είναι ότι θα μπορούσε να λειτουργήσει ως το θεμέλιο για επιπρόσθετους φυσικούς τελεστές σε συστήματα όπως το Pig για πιο αποτελεσματική επεξεργασία γράφων.

Με την διευρυμένη προσαρμογή του Semantic Web σε διάφορους τομείς όπως επιχειρήσεις, χρηματοοικονομικά, βιοπληροφορική και επιστημονική έρευνα υπάρχει μια αυξανόμενη ανάγκη για αποτελεσματικές τεχνικές για επεξεργασία και αναζήτηση των δεδομένων RDF. Το επίκεντρο αλλάζει από απλές αναζητήσεις ταιριάσματος των προτύπων οι οποίες περιλαμβάνουν ένα επιλέγω – σχεδιάζω - ενώνω σε πιο αναλυτικές αναζητήσεις που περιλαμβάνουν περίπλοκες ομαδοποιήσεις και αθροίσματα. Οι δημοφιλής γλώσσα αναζήτησης για το RDF SPARQL [10], αυτήν την στιγμή δεν χρησιμοποιεί τελεστές ομαδοποίησης και αθροίσματος αλλά η επόμενη έκδοση θα συμπεριλαμβάνει αυτές τις κατασκευές ως απάντηση στις αυξανόμενες απαιτήσεις των χρηστών. Πριν επεξεργαστούμε την RDF SPARQL, θα επανεξετάσουμε άλλες προσπάθειες σχετικές με την εργασία μας.

8.1 Σχετικές Εργασίες

Τα παραδοσιακά συστήματα OLAP υποστηρίζουν αποτελεσματικά αναλυτικές αναζητήσεις αλλά εστιάζουν σε δομημένα δεδομένα τα οποία έχουν οργανωθεί

κατάλληλα σε σχήμα αστεριού ή νιφάδας χιονιού(snowflake).Τα περισσότερα από αυτά τα εξελιγμένα συστήματα είναι πολύ ακριβά. Συστήματα παράλληλης επεξεργασίας που βασίζονται σε μηχανές όπως το Google Mapreduce[1] γίνονται όλο και πιο δημοφιλή χάρη στην φύση τους που είναι κλιμακωτή και την αποτελεσματικότητα σε σχέση με τα κόστη που τους δίνει ένα πλεονέκτημα σε σχέση με τα εξελιγμένα συστήματα OLAP. Το μοντέλο προγραμματισμού MapReduce[1] εισήχθη για να καλύψει τις αυξανόμενες ανάγκες για ανάλυση ad-hoc δεδομένων σε πολύ μεγάλα σύνολα δεδομένων. Ωστόσο, αυτό το πλαίσιο έχει σχεδιαστεί να δουλεύει με ένα ομογενές σύνολο δεδομένων και έτσι δεν υποστηρίζει αποτελεσματικά σχεσιακές λειτουργίες όπως τα Joins. Έχουν προταθεί επεκτάσεις για επεξεργασία δομημένων δεδομένων. Μια τέτοια προσέγγιση Map-Reduce-merge[4] προσπαθεί να ξεπεράσει αυτό το πρόβλημα προσθέτοντας μια φάση συγχώνευσης στο MapReduce το οποίο θα μπορεί αποτελεσματικά να συγχωνεύσει δεδομένα τα οποία έχουν ήδη διαμεριστεί και κατηγοριοποιηθεί από τα πρότυπα Map και Reduce. Επίσης το MapReduce είναι πολύ χαμηλού επιπέδου και απαιτεί από τους χρήστες να γράφουν προγράμματα σε διαδικαστικό στυλ, το οποίο είναι ¨μπελάς¨ και δεν επιτρέπει αυτόματη βελτιστοποίηση. Έτσι κάποιες υψηλού επιπέδου γλώσσες όπως Pig Latin[2], SCOPE[5], DryadLINQ[6], Sawzall[7] και JAQL[13] έχουνε προταθεί για την επίλυση αυτού του προβλήματος. Μια άλλη κατεύθυνση προς την βελτίωση συμπεριλαμβάνει μια νέα υβριδική προσέγγιση HadoopDB[8] η οποία συνδυάζει τα πλεονεκτήματα επίδοσης και αποτελεσματικότητας των παράλληλων βάσεων δεδομένων με τα πλεονεκτήματα κλιμάκωσης και ανοχής λαθών των συστημάτων MapReduce. Το HadoopDB διαχωρίζει τα δεδομένα όπως απαιτείται από την αναζήτηση και εισάγει όσο το δυνατόν περισσότερες λειτουργίες στην βάση δεδομένων. Το πλεονέκτημα της εισαγωγής υπολογισμών στην βάση δεδομένων περιορίζεται στην επεξεργασία της πρώτης εργασίας reduce το οποίο συνήθως σημαίνει το τέλος της πρώτης λειτουργίας join.

Η επεξεργασία των δεδομένων RDF συνήθως απαιτεί διάφορα joins και πράξεις ομαδοποίησης, το οποίο δεν μπορεί αποτελεσματικά να εισαχθεί στην βάση δεδομένων. Μια ακόμη προσέγγιση βελτιστοποιεί τα πολλών κατευθύνσεων joins παρέχοντας στρατηγικές για να διαχωρίζει και να αναπαράγει τις πλειάδες μιας σχέσης σε διαδικασίες του reducer με έναν τρόπο που ελαχιστοποιεί το κόστος επικοινωνίας. Αυτή η εργασία είναι συμπληρωματική στην δική μας προσέγγιση και ενσωματώνοντας το σύστημα διαχωρισμού μέσα στο Pig, μπορούμε να βελτιώσουμε περισσότερο την επίδοση των λειτουργιών join. Η RDF κοινότητα έχει πρόσφατα αποδεχθεί το παράδειγμα επεξεργασίας παράλληλων δεδομένων όπως περιγράφηκε από το μοντέλο MapReduce και έχουν γίνει προσπάθειες να εκτελέσουμε συλλογιστική κλιμάκωση RDF, υλοποιώντας την κλείστοτητά του σχετικού γράφου και έτσι να εκτελέσει αποτελεσματική συλλογιστική χρησιμοποιώντας την σειρά που προκύπτουν από τους κανόνες εξαγωγής συμπεράσματος. Έχουν υπάρξει προσεγγίσεις βασιζόμενες στο MapReduce για ταίριασμα προτύπου, αποσυνθέτοντας τα γραφήματα σε μόρια RDF.

Ωστόσο αυτές οι τεχνικές ίσως να μην είναι αποτελεσματικές για αναλυτικές αναζητήσεις που απαιτούν περίπλοκες ομαδοποιήσεις και αθροίσματα. Το RAPID[9], προτείνει μια γλώσσα υψηλού επιπέδου που επεκτείνει το Pig για να συμπεριλάβει αλγορίθμους για να εκφράσει περίπλοκες αναλυτικές αναζητήσεις και έναν τελεστή

υβριδικού join που βελτιστοποιεί τις διάφορες ομαδοποιήσεις και αθροίσματα. Επικεντρωνόμαστε σε αναλυτικές αναζητήσεις αλλά εστιάζουμε στο να βελτιώσουμε την φάση ταιριάσματος του προτύπου. Κατασκευάζουμε πάνω στο Pig ένα σύστημα επεξεργασίας παράλληλων δεδομένων που στοχεύει στο Hadoop και παρόμοια περιβάλλοντα εκτέλεσης. Το Pig προσφέρει υψηλού επιπέδου γλώσσα, Pig Latin, το οποίο υποστηρίζει επεξεργασία μεγάλων ad-hoc συνόλων δεδομένων. Παρουσιάζουμε μια επέκταση του RAPID το οποίο ονομάζουμε RAPID+. Ειδικότερα κάνουμε τις ακόλουθες συνεισφορές :

- Προτείνουμε (ξαναπαραγοντοποιώντας) αναλυτική επεξεργασία σε γραφήματα RDF με νέους τελεστές ομαδοποίησης που επιτρέπουν πιο παραλληλοποιημένη επεξεργασία. Επίσης περιορίζει τον απαιτούμενο αριθμό των ενδιάμεσων βημάτων και έτσι μειώνει το συνολικά κόστη I/O.
- Παρουσιάζουμε μια προσέγγιση πρόβλεψης. Εννοούμε ότι πρέπει να ενώσουμε σε αυτήν την φάση υπολογισμούς οι οποίοι θα πρέπει συνήθως να είναι μέρος της μεταγενέστερης φάσης με ένα τρόπο που ελαχιστοποιεί τα συνολικά κόστη των δύο φάσεων.
- Παρουσιάζουμε αποτελέσματα αξιολόγησης σε ένα συνθετικό σύνολο δεδομένων που παράγεται από μια γεννήτρια σημείου αναφοράς.

8.2 Υπόβαθρο και κίνητρο

8.2.1Επεξεργασία δεδομένων στο Pig

Το μοντέλο δεδομένων του Pig Latin αποτελείται από μια ατομική τιμή, μια πλειάδα (καταγραφές δεδομένων που έχουν μια συνοχή από πεδία όπου κάθε πεδίο μπορεί να είναι ένα άτομο, μια πλειάδα, ένας σάκο δεδομένων) και ένα σάκο δεδομένων ο οποίος είναι μια συλλογή από πλειάδες. Έχει πρωτόγονες λειτουργίες όπως το load για να διαβάσει το εισαγόμενο έγγραφο ,FOREACH λειτουργία η οποία χρησιμοποιείται ως ένας επαναληπτικός βρόχος μιας συλλογής,FILTER για να φιλτράρει πλειάδες ,λειτουργίες ομαδοποίησης όπως GROUP,και βασικές λειτουργίες αθροίσματος όπως SUM,MAX,COUNT,AVG.Η συσχέτιση ανάμεσα σε δύο πίνακες με σχέσεις A και B στην στήλη 0 μπορεί να εκφραστεί χρησιμοποιώντας την λειτουργία Join ως εξής :

JOIN A by $0, B by $0;

Το Pig Latin έχει επίσης μια ειδική λειτουργία ομαδοποίησης η οποία ονομάζεται COGROUP η οποία μπορεί να ομαδοποιήσει από κοινού τις πλειάδες δυο ή περισσοτέρων σχέσεων. Για να χειριστούμε περιπτώσεις όπου μια σχέση χρειάζεται να επεξεργαστεί με διάφορους τρόπους αλλά χωρίς να επιβαρυνθούμε με το κόστος φόρτωσής της πολλές φορές, μπορούμε να χρησιμοποιήσουμε τον τελεστή split ο οποίος χωρίζει τα περιεχόμενα μιας σχέσης σε δύο ή περισσότερες εκφράσεις υπό συνθήκη. Για παράδειγμα μια σχέση A με ηλικία χαρακτηριστικών όπως στην στήλη 1, μπορεί να διαχωριστεί βασιζόμενη στην τιμή της ηλικίας με ελάχιστο και μέγιστο:

SPLIT A into minors IF $1 < 18;
Majors IF $1 >=18;

Επιπρόσθετη λειτουργικότητα μπορεί να επιτευχθεί μέσω των UDF's(user defined functions). Οι ερωτήσεις σε PigLatin συγκεντρώνονται σε μια ακολουθία MapReduce εργασιών που τρέχουν πάνω από το Hadoop[11], μια ανοιχτού κώδικα υλοποίηση του MapREDUCE. Στον προγραμματισμό στο μοντέλο MapReduce οι εργασίες κωδικοποιούνται χρησιμοποιώντας δυο ανεξάρτητες λειτουργίες : την map και την reduce ,και η εκτέλεσή τους, μπορεί να παραλληλιστεί σε ένα cluster από μηχανήματα. Η συνάρτηση map διαβάζει κάθε γραμμή των δεδομένων και την μετατρέπει σε ένα ζευγάρι της μορφής <key,value>. Όταν όλες οι εργασίες του mapper έχουν ολοκληρωθεί, τα ενδιάμεσα κλειδιά έχουν ταξινομηθεί και συγχωνευτεί. Τα αντικείμενα με το ίδιο κλειδί για παράδειγμα {key,list(values)},συγκεντρώνονται σε έναν συγκεκριμένο reducer .Η συνάρτηση reduce στοχεύει στην λειτουργία αθροίσματος για αυτά τα δεδομένα. Ορισμένες φορές, μια συνδυαστική(combiner) συνάρτηση μπορεί να χρησιμοποιηθεί για να παρέχει μερικά αθροίσματα στην φάση του map.

8.3 Αναλυτική επεξεργασία των δεδομένων RDF στο Pig

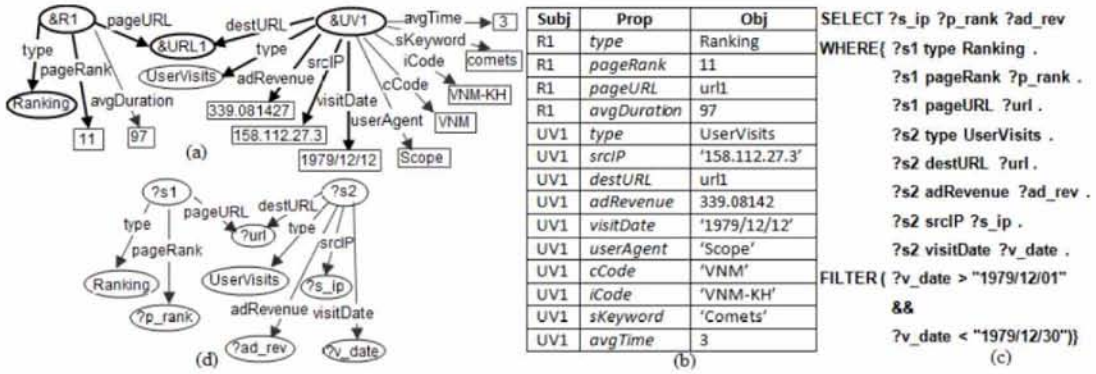
Στο γενικότερο πλαίσιο της αναλυτικής επεξεργασίας των δεδομένων RDF υπάρχουν μερικές προκλήσεις που προέρχονται από τα χαρακτηριστικά του μοντέλου του. Μια βάση δεδομένων RDF είναι μια συλλογή από statements, όπου κάθε μια είναι μια τριπλέτα του τύπου<subject,property,object>, το οποίο υποδηλώνει ότι ένα subject σχετίζεται με μια property της οποίας η αξία υποδηλώνεται από το object. Η εικόνα 1(a)δείχνει έναν κατευθυνόμενο γράφημα αυτών των statements συνδέοντας τους κόμβους-πηγές ((&R1, &UV1 and &URL1) και τις τιμές των ιδιοτήτων μέσω ακμών(με κάποια τιμή) που αντιπροσωπεύουν ιδιότητες. Το μοντέλο RDF στην εικόνα 1(a) απεικονίζει ένα πραγματικό σενάριο των αρχείων καταγραφής του διακομιστή HTTP και χρησιμοποιείται σαν ένα παράδειγμα για το υπόλοιπο paper. Στο παράδειγμά μας, η ρίζα **&R1** έχει τρεις ιδιότητες : pagerank με ακέραια τιμή 11, avgDuration με τιμή 97, και την ιδιότητα pageURL με ρίζα αυτής της τιμής &URL 1. Παρουσιάζουμε τριπλέτες ως τριαδικές σχέσεις όπως φαίνεται στην εικόνα 1b. Σημειώστε ότι διαφορετικά από τα παραδοσιακά OLAP ΣΥΣΤΗΜΑΤΑ,ΤΑ ΔΕΔΟΜΈΝΑ RDF δεν οργώνονται σε σχήμα αστεριού ή snowflake και θα απαιτούσαν star joins και chain joins για να επανασυναρμολογήσουν τις δυαδικές σχέσεις σε ν-δικές σχέσεις ισοδύναμες με τους παράγοντες και την διάσταση πλειάδων σε ένα σχεσιακό σχήμα.

Η θεμελιώδης κατασκευή ερωτήματος στο SPARQL είναι ένα πρότυπο ταιριάσματος σε μορφή γραφήματος το οποίο είναι ισοδύναμο με το select-project-join στην SQL. Η εικόνα 1c δείχνει ένα παράδειγμα ταιριάσματος προτύπου. Το γράφημα αποτελείται από τρια πεδία για παράδειγμα ("?s1 pageURL ?url)όπου τα ονόματα μεταβλητών έχουν ως πρόθεμα το ? και μπορούν να χρησιμοποιηθούν για να αντιπροσωπεύσουν τα Subject,property και Object. Η εικόνα 1d αντιπροσωπεύει το γράφημα που αντιστοιχεί στην αναζήτηση στην εικόνα 1c,όπου οι μεταβλητές υποδηλώνονται ως κόμβοι στο γράφημα. Ο στόχος είναι να υπερθέσουμε αυτό το

μοντέλο στο γράφημα δεδομένων της εικόνας 1a και να βρούμε τα αντίστοιχα υπογραφήματα τα οποία αντιστοιχούν στις ταμπέλες και στις ακμές που φαίνονται με σκούρα γράμματα στην εικόνα 1a. Προτάσεις είναι σε εξέλιξη για την επέκταση του SPARQL που θα περιλαμβάνει δομές ομαδοποίησης και αθροίσματος που θα επιτρέπουν μια αναλυτική αναζήτηση όπως "Υπολόγισε τον μέσο όρο PageRank και τον συνολικό adRevenue για τις σελίδες που έχουν επισκεφτεί από ένα συγκεκριμένο srcIP με ημερομηνία επίσκεψης ανάμεσα σε 1/12/1979 και 30/12/1979".

Αυτό μπορεί να θεωρηθεί οτι αποτελείται από τρεις βασικές δομές αναζήτησης : Πρώτον: χρήση ταιριάσματος προτύπων για τον υπολογισμό υπογράφων που αντιστοιχούν στις σελίδες που έχει επισκεφτεί ένα συγκεκριμένο srcIP με μια συνθήκη φιλτραρίσματος στην τιμή της ημερομηνίας επίσκεψης visitDate(i.e. visitDate > 1979/12/01 and visitDate < 1979/12/30)
Δεύτερον: ομαδοποίηση των προτύπων των αποτελεσμάτων και τρίτον το άθροισμα των τιμών του pageRank και adRevenue .

Η παραπάνω αναζήτηση έχει δυο πρότυπα σε σχήμα αστεριού: Πρώτον τριπλέτες που αναπαριστούν πόρους του τύπου Ranking με ιδιότητες pageURL and pageRank. Και δεύτερον τριπλέτες που αναπαριστούν πόρους του τύπου UserVisits με ιδιότητες srcIP, destURL, adRevenue, και visitDate. Επιπλέον, έχουμε ένα ακόμα join για να συνδέσουμε τα δυο αστέρια μέσω της ιδιότητας destURL and pageURL. Στην πραγματικότητα οι πιο αναλυτικές αναζητήσεις έχουν πολλά joins, πολλές ομαδοποιήσεις και αθροίσματα. Στο [15] βελτιστοποιήσαμε το τελευταίο και τώρα εστιάζουμε στην φάση αντιστοίχισης του μοντέλου.



Εικόνα 1:

a)Παράδειγμα μοντέλου RDF

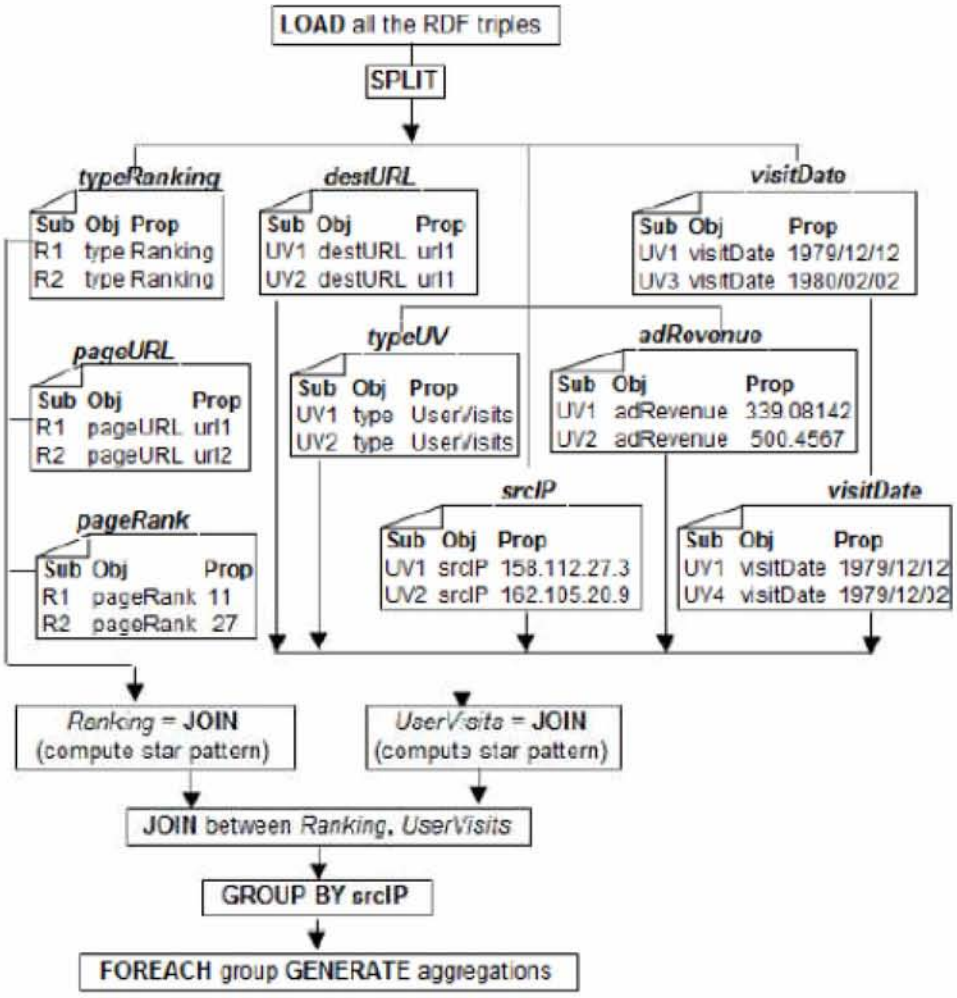
b)RDF τριπλέτα σαν τριαδικές σχέσεις

c)Παράδειγμα ταιριάσματος προτύπου σε SPARQL

d)Πρότυπο γράφου

Πηγή:P.Ravindra,V.Deshpande,K.Anyanwu,Towards Scalable RDF Graph Analytics on MapReduce Atbrox.com/about

8.4Ταίριασμα προτύπων



2 α)Επεξεργασία παραδείγματος ερώτησης χρησιμοποιώντας τελεστές Pig

εικόνα 2β:Υπολογισμός ranking

Πηγή:P.Ravindra,V.Deshpande,K.Anyanwu,Towards Scalable RDF Graph Analytics on MapReduce

Το παράδειγμα αναζήτησης είναι ένα απλό πρότυπο γράφου που αποτελείται από δύο πρότυπα αστεριού(star).Υπάρχουν δυο προσεγγίσεις για τον υπολογισμό των προτύπων αστεριού και την σύνδεση μεταξύ τους. Η πρώτη προσέγγιση βασίζεται σε ένα μοντέλο τριπλής αποθήκευσης παρόμοιο με την εικόνα 1 β. Σε παραδοσιακά σχεσιακά συστήματα ένα πρότυπο αστεριού(star) σε μια τέτοια τριπλή σχέση υπολογίζεται σταδιακά με κάθε ένωση να προσθέτει μια ακμή. Στο Pig μια n-τρόπων ένωση ανάμεσα σε σχέσεις που ενώνονται στην την ίδια στήλη μπορεί να υπολογιστούν με έναν μονό κύκλο mapreduce. Αυτό έχει ως αποτέλεσμα ένα σύστημα αξιολόγησης στο οποίο κάθε ένωση αστεριού μπορεί να υπολογιστεί σαν μια ένωση ν κατευθύνσεωνμε από ν αντίγραφα της τριπλής σχέσης. Για παράδειγμα το πρότυποαστεριού που αντιπροσωπεύει πηγές του τύπου Ranking και ιδιότητες PageRank και Pageurl μπορούν να υπολογιστούν χρησιμοποιώντας μια λειτουργία JOIN τριων κατευθύνσεων στα αντίγραφα της σχέσης εισόδου για παράδειγμα τριπλέτα 1,2 και 3 ως :

starPattern = JOIN triples1 ON Sub,
triples2 ON Sub, triples3 ON Sub;

Εδώ και λίγο καιρό η λειτουργία join στο Pig δεν εφαρμόζεται σαν μια φυσική ένωση που έχει ως αποτέλεσμα παραπανήσιες στήλες. Αυτός ο πλεονασμός μπορεί να μειωθεί προβάλλοντας μόνο τις απαιτούμενες στήλες μετά από κάθε join.

Η δεύτερη προσέγγιση χρησιμοποιεί μοντέλο αποθήκευσης που αναφέρεται ως vertical partitioning[3]. Αυτό έχει ως αποτέλεσμα μικρότερες σχέσεις που περιέχουν πλειάδες με τον ίδιο τύπο ιδιότητας οι οποίες μπορούν να επιτευχθούν χρησιμοποιώντας την λειτουργία split του Pig όπως φαίνεται στην εικόνα 2α. Για να αξιολογήσουμε το πρότυπο γράφου χρησιμοποιώντας αυτό το μοντέλο αποθήκευσης, οι σχέσεις split χρειάζεται να ενωθούν για να σχηματίσουν πρότυπα αλυσίδας ή αστεριού. Η προσέγγιση vertical partitioning έχει τα ακόλουθα ζητήματα

-Κάθε συνθήκη split οδηγεί σε μια ταυτόχρονη υπορροή η οποία ανταγωνίζεται για πηγές μνήμης, οδηγούμενη στην πιθανότητα διαρροών δίσκου και αυξημένων τιμών κόστους εισόδου-εξόδου.
-Ένα άλλο θέμα αφορά την δομή των ενδιάμεσων σχέσεων, σε αντίθεση με το σχήμα των σχέσεων του οποίου οι στήλες είναι ονόματα ιδιοτήτων, το σχήμα από όλα αυτά τα τμήματα είναι του τύπου (Sub, Prop, Obj) όπως μπορούμε να δούμε στην εικόνα 2α. Ενώ το Pig καταγράφει στήλες με την σειρά που καταγράφονται με την εντολή join, όπως μπορούμε να δούμε στην εικόνα 2β ,χρειαζόμαστε να διατηρήσουμε μια χαρτογράφιση ανάμεσα σε αυτά τα τμήματα και τις ιδιότητες(properties) που εμπεριέχονται σε αυτά. Επιπλέον για κάθε λειτουργία ένωσης/ομαδοποίησης και αθροίσματος είναι απαραίτητο να κρατάμε λογαριασμό της σειράς με την οποία τα τμήματα ενώνονταν στις προηγούμενες λειτουργίες ή να εφεύρουμε μια συνθήκη ονοματοδοσίας η οποία μοναδικά θα μπορεί να αναγνωρίσει κάθε στήλη του κάθε τμήματος.

Είναι σημαντικό να σημειώσουμε ότι στην προσέγγιση Pig κάθε μια από αυτές τις ενώσεις αστεριού αντιστοιχεί σε ένα ξεχωριστό κύκλο mapreduce. Επεξεργαζόμενοι αναλυτικές αναζητήσεις σε δεδομένα RDF συναντούμε διάφορες τέτοιες ενώσεις

αστεριού που οδηγούν στην υλοποίηση ενδιάμεσων αποτελεσμάτων στην φάση map και reduce κάθε κύκλου mapreduce και έτσι αυξάνουμε τα κόστη I/O. Στην πραγματική ζωή χρειάζεται να αντιμετωπίσουμε ακόμα πιο πολύπλοκες αναζητήσεις που απαιτούν διάφορες ενώσεις ,πολλαπλές ομαδοποιήσεις και αθροίσματα. Συνοψίζοντας ,το Pig Latin βασίζεται σε σχεσιακές αλγεβρικές πλειάδες των οποίων οι τελεστές δεν είναι τόσο αποτελεσματικοί για την επεξεργασία πολύπλοκων αναλυτικών αναζητήσεων στο RDF. Όπως έχει καταγραφεί παραπάνω, η επεξεργασία δεδομένων που αναπαριστώνται σε γράφημα χρησιμοποιώντας τις υπάρχουσες λειτουργιές του Pig έχει το ρίσκο υψηλών κοστών εισόδου εξόδου που είναι αποτέλεσμα πολλαπλών αυτοενώσεων και υλοποίηση των ενδιάμεσων αποτελεσμάτων σε κάθε κύκλο mapreduce που αντιστοιχούν (τα αποτελέσματα) σε κάθε απαιτούμενη ένωση. Στο επόμενο τμήμα παραγοντοποιούμε τις κοινές προκλήσεις στην επεξεργασία RDF για αναλυτικές αναζητήσεις και παρουσιάζουμε μια προσέγγιση που μας βοηθά να ξεπεράσουμε κάποιες από αυτές τις αδυναμίες.

```
input = LOAD '\data' using
loadFilter ( pageRank, pageURL,
             type:Ranking,
             destURL, adRevenue,
             srcIP, visitDate,
             type:UserVisits )
```

εικόνα 3α

Sub	Prop	Obj
R1	*type*	Ranking
R1	*pageRank*	11
R1	*pageURL*	url1
UV1	*type*	UserVisits
UV1	*srcIP*	158.112.27.3
UV1	*destURL*	url1
UV1	*adRevenue*	339.08142
UV1	*visitDate*	1979/12/12

3β

Παράδειγμα φιλτραρίσματος μη σχετικών τριπλετών

Πηγή :P.Ravindra,V.Deshpande,K.Anyanwu,Towards Scalable RDF Graph Analytics on MapReduce

Η διαίσθηση πίσω από κάθε προσέγγισή μας είναι να εκμεταλλευτούμε το γεγονός ότι τα RDF δεδομένα αποτελούνται από μια σειρά προτύπων αστεριού και αλυσίδας της τελευταία κατηγορία. Μπορούν να επεξεργαστούν ταυτόχρονα χρησιμοποιώντας έναν αλγόριθμό βασισμένο στην ομαδοποίηση. Ο στόχος μας είναι να ελαχιστοποιήσουμε τα κόστη εισόδου και εξόδου στοχεύοντας την πηγή του προβλήματος. Για παράδειγμα ο διαδοχικός υπολογισμός ξεχωριστών προτύπων αστεριού. Τα κόστη εισόδου εξόδου μειώνονται επίσης χρησιμοποιώντας δυο στρατηγικές για την εκτέλεση λειτουργίας: operator-coalescing και look-ahead processing. Οι λειτουργίες Rapid+ πρόσφατα υλοποιήθηκαν ως Pig Latin udfs.

8.5 Τελεστής Συνένωση(loadfilter)

Τα δεδομένα εισόδου RDF αποτελούνται από μια συλλογή τριπλών εντολών. Είναι χρήσιμο να κόψουμε όσες περισσότερες άσχετες τριπλέτες όσο το δυνατόν πιο νωρίς στην επεξεργασία. Στο παράδειγμά μας οι μόνες ιδιότητες που είναι σχετικές στην αναζήτησή μας έχουν μαρκαριστεί με σκούρα γράμματα στην εικόνα 1α, συνεπώς αντί να χρησιμοποιούμε ένα τελεστή load ακολουθούμενο από ένα τελεστή filter χρησιμοποιούμε την λειτουργία load filter η οποία συνενώνει τις δυο λειτουργίες. Αυτό μειώνει το κόστος εναλλαγής περιβάλλοντος(context swi..)από την μια μέθοδο στην άλλη περνώντας τις παραμέτρους ανάμεσα σε μεθόδους και πολλαπλούς χειρισμούς των ίδιων δεδομένων σε δυο φάσεις συνενώνοντας την φάση φιλτραρίσματος και φόρτωσης. Η συνάρτηση loadfilter χρησιμοποιείται σαν μια παράμετρος του τελεστή PigLatin load(εικόνα 3α).

Ένα σημείο που αξίζει να αναφερθούμε είναι ότι διαφορετικά από την στρατηγική κατακόρυφου διαμερισμού που παρουσιάστηκε προηγουμένως χρησιμοποιώντας το PigLatin το αποτέλεσμα της συνάρτησης loadfilter είναι μια σχέση που περιλαμβάνει όλες τις σχετικές τριπλέτες, ασχέτως από τον τύπο property. Η εικόνα 3β δείχνει το αποτέλεσμα της εφαρμογής της συνάρτησης loadfilter στο παράδειγμα σχέσης της τριπλέτας όπως φαίνεται στην εικόνα 1β.

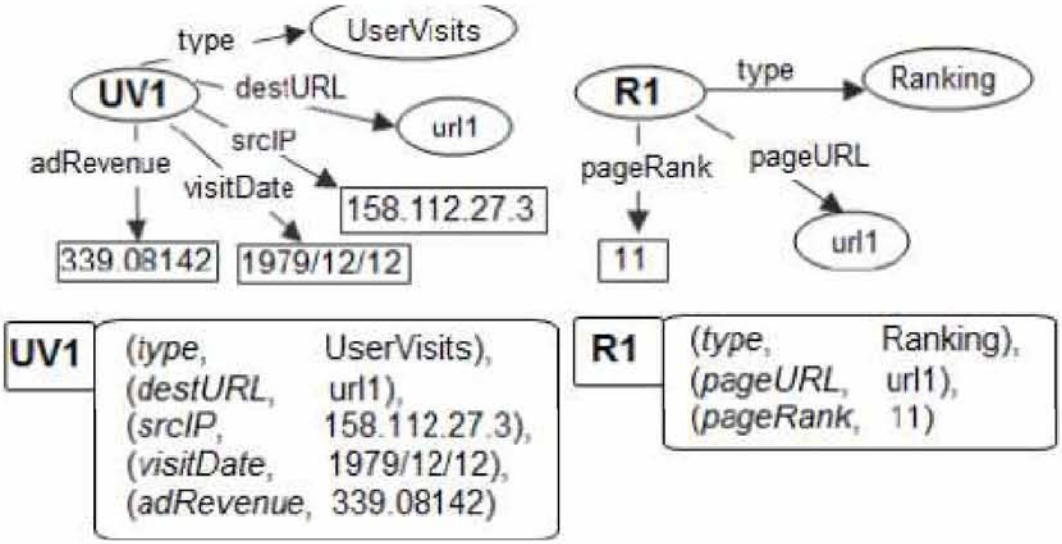
8.6Ταίριασμα προτύπου

8.6.1Υπολογισμός ένωσης αστεριού που βασίζεται στην ομαδοποίηση

Τα υπογραφήματα αστεριού που ταιριάζουν με τα πρότυπα(στην περίπτωσή μας τα πρότυπα ranking και uservisits)μπορούν υπολογιστούν ταυτόχρονα χρησιμοποιώντας μια στρατηγική επεξεργασίας βασισμένη στην συσταδοποίηση. Για παράδειγμα, χρησιμοποιώντας τον τελεστή GROUPBY του Pig ,μπορούμε να ομαδοποιήσουμε τις τριπλέτες που βασίζονται στην στήλη subject και να υπολογίσουμε όλα τα υπογραφήματα με το ίδιο subject. Η εικόνα 4 δείχνει ένα στιγμιότυπο του αποτελέσματος με μια ομάδα το καθένα από την τριπλέτα ranking και uservisits. Αυτό μπορεί να επιτευχθεί με έναν μοναδικό κύκλο mapreduce, αντίθετα με την προσέγγιση pig με την στρατηγική vertical partitioning, η οποία υπολογίζει ξεχωριστά πρότυπα αστεριού σε διαφορετικούς κύκλους mapreduce. Ωστόσο υπάρχουν κάποια θέματα με αυτήν την προσέγγιση :

-Αυτό έχει ως αποτέλεσμα σε ετερογενής σάκους που περιέχουν πλειάδες που αντιστοιχούν στις κορυφές στο αστεροειδές πρότυπο. Αυτοί οι σάκοι σχολιάζονται με το subject όπως μπορούμε να δούμε στην figure 4. Καθώς όλες αυτές οι πλειάδες έχουν το ίδιο πεδίο subject, αυτό το πεδίο είναι άφθονο ανάμεσα στις πλειάδες. Έτσι, βελτιώνουμε την αποτελεσματικότητα χώρου κρατώντας πεδία(property,object) και μειώνοντας τα παραπανήσια πεδία(αυτά που δεν χρειαζόμαστε).

-Αυτή η προσέγγιση υπολογίζει όλα τα πιθανούς υπογράφους αστεριού περιλαμβάνοντας αυτούς οι οποίοι ίσως να μην ταιριάζουν στο απαιτούμενο πρότυπο εξαιτίας στοιχείων που λείπουν, π.χ. ακμές. Έτσι, χρειαζόμαστε να φιλτράρουμε και να κρατήσουμε μόνο αυτούς τους υπογράφους που είναι σχετικοί στην αναζήτησή μας. Εκτελούμε ένα φιλτράρισμα που βασίζεται στην δομή για να μειώσουμε πρότυπα τα οποία είναι δομικά ημιτελή. Επιπλέον εκτελούμε φιλτράρισμα βασισμένο στην τιμή,το οποίο επικυρώνει κάθε υπογράφο αστεριού. Για παράδειγμα visitDate property i.e. VisitDate > 1979/12/01 and visitDate < 1979/12/30.
Σημειώστε ότι οι πλειάδες που αντιστοιχούν στις ιδιότητες φιλτραρίσματος δεν απαιτούνται πια στις μεταγενέστερες φάσεις. Έτσι τις μειώνουμε για να βελτιώσουμε την αποτελεσματικότητα χώρου.



εικόνα 4:υπολογισμός προτύπου αστεριού χρησιμοποιώντας Group By subject

Πηγή:P.Ravindra,V.Deshpande,K.Anyanwu,Towards Scalable RDF Graph Analytics on MapReduce

8.6.2 Σύνδεση αστεριών και προβλέψιμη επεξεργασία

Κανονικά η επόμενη φάση θα ήταν να υπολογίσει το join ανάμεσα στα δυο υπογραφήματα αστεριού. Στην περίπτωση των κανονικών συνδέσεων, έχουμε επισημάνει πλειάδες όπου μπορούμε να προσδιορίσουμε την μεταγενέστερη λειτουργία σε ένα συγκεκριμένο πεδίο. Στην περίπτωση της στρατηγικής που βασίζεται στην ομαδοποίηση για να υπολογίσουμε τα αστέρια, έχουμε ετερογενής σάκους που περιέχουν πλειάδες χωρίς ετικέτα, με διαφορετικές τιμές property-object. Για να βρούμε την τιμή μιας συγκεκριμένης ιδιότητας(property),πρέπει να σκανάρουμε ολόκληρο τον σάκο. Λαμβάνουμε υπόψιν μας το σενάριο που η επόμενη φάση είναι μια σύνδεση ανάμεσα στους υπογράφους αστεριού. Χρειάζεται να επεξεργαστούμε αυτούς τους σάκους για

ακόμη μια φορά στην μεταγενέστερη φάση map, για να τα σχολιάσουμε βασιζόμενοι στα κλειδιά συνένωσης. Για το παράδειγμα αναζήτησης, θα επεξεργαζόμασταν καθέναν από τους σάκους στην φάση reduce της λειτουργίας group-by για να φιλτράρουμε την τιμή του visitdate. Μετά, στην φάση map της μεταγενέστερης λειτουργίας σύνδεσης, θα επεξεργαζόμασταν ξανά καθέναν από αυτούς τους σάκους για να προσδιορίσουμε τις τιμές των ιδιοτήτων σύνδεσης .π.χ. PageURL και destURL.Για να ελαχιστοποιήσουμε αυτού του είδους την επαναλαμβανόμενη επεξεργασία των σάκων, μπορούμε να προβλέψουμε τον τελεστή ο οποίος θα βοηθήσει στην προετοιμασία των δεδομένων για την επόμενη φάση. Όπως γίνεται με τις πλειάδες που αντιστοιχούν στην ιδιότητα φιλτραρίσματος, μειώνουμε επίσης τις πλειάδες που αντιστοιχούν στις ιδιότητες σύνδεσης(destURL και destURL)για να μειώσουμε την χρήση χώρου. Το ακόλουθο είναι ένα παράδειγμα RAPID+UDF που υλοποιεί την διαδικασία πρόβλεψης(lookahead) κατά την διάρκεια της φάσης ομαδοποίησης-φιλτραρίσματος:

```
starSubgraphs = GROUP input BY $0;
reassembled = FOREACH starSubgraphs GENERATE
      reassemble($1,fc,'prop1','prop2|prop3')
```

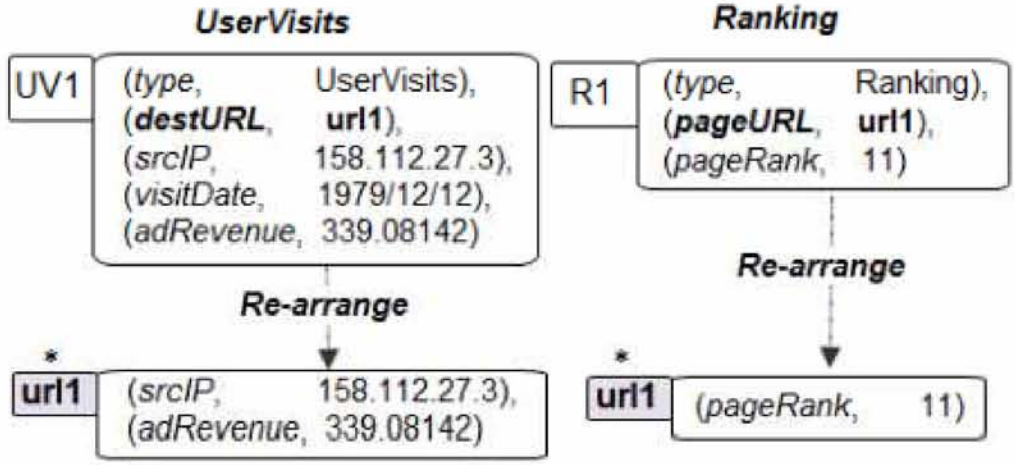
Ο ψευδοκώδικας 1 αντιπροσωπεύει την επεξεργασία σύνδεσης lookahead κατά την διάρκεια της φάσης ομαδοποίησης-φιλτραρίσματος. Στις γραμμές 2-5 του ψευδοκώδικα αξιολογούμε την συνθήκη του φίλτρου ανάλογα με την τιμή της ιδιότητας του φίλτρου. Αν η συνθήκη αποτύχει μειώνουμε ολόκληρο τον σάκο. Καθώς η επόμενη φάση περιλαμβάνει μια σύνδεση, θέλουμε να σχολιάσουμε τον σάκο που βασίζεται στο κλειδί της ένωσης.

Pseudocode 1 : Join look-ahead during the group-filter

```
Input   :Bag b, filter condition fc on prop1,
         Join on (prop2 or prop3)
Output :Re-arranged bag b or null
1: for all Tuple t ∈ b do
2:     if t.Property = prop1 then
3:         if fc (t.Object) = FALSE then
4:             return null
5:         end if
6:         if t.Property = (prop2 or prop3) then
7:             Set b.key = t.Object
8:         end if
9:         if (t.Property = type then
10:            Set className=t.Object
11:           if (prop2 or prop3) = type:className then
12:               Set b.key = t.Subject
13:           end if
14:        end if
15:        Remove t from b
16:    end if
17: end for
18: return b
```

Πηγή:P.Ravindra,V.Deshpande,K.Anyanwu,Towards Scalable RDF Graph Analytics on MapReduce

Υπάρχουν δύο σενάρια :α)μπορεί να θέλουμε να συνδέσουμε(join) βασιζόμενοι στην τιμή μιας ιδιότητας(π.χ. Dest URL) όπως χειρίζονται οι γραμμές 6-8 ή β)η σύνδεση(join) μπορεί να συνδέει ένα στιγμιότυπο μιας συγκεκριμένης τάξης(π.χ. Type:Ranking),το οποίο πραγματοποιείται από τις γραμμές 9-14.Για παράδειγμα η εικόνα 5 δείχνει ένα στιγμιότυπο μετά την επεξεργασία του lookahead κατά την διάρκεια της φάσης ομαδοποίηση-φιλτράρισμα(group-filter),όπου η σάκοι σχολιάζονται με μια νέα τιμή url1 η οποία αντιστοιχεί στην τιμή των ιδιοτήτων σύνδεσης π.χ. DestURL και pageURL.Η επόμενη φάση είναι να υπολογίσουμε την πραγματική σύνδεση ανάμεσα στα πρότυπα αστεριού και ακολουθείται από μια λειτουργία ομαδοποίησης που βασίζεται στην τιμή της ιδιότητας srcIP,και ακολουθεί μια πρόσθεση των υπολογισμένων ομάδων. Για ακόμα μια φορά επωφελούμαστε από τον τελεστή lookahead κατά την διάρκεια της φάσης σύνδεσης για να επανασυναρμολογήσουμε κάθε σάκο και να τον προετοιμάσουμε για την φάση ομαδοποίησης π.χ. Να το σχολιάσουμε με την τιμή της ιδιότητας ομαδοποίησης (srcIP).



εικόνα 5:Παράδειγμα look-ahead processing

Πηγή:P.Ravindra,V.Deshpande,K.Anyanwu,Towards Scalable RDF Graph Analytics on MapReduce

8.7. Μελέτη περιπτώσεων

Σε αυτόν τον τομέα ,αξιολογούμε την προσέγγισή μας συγκρίνοντάς την με την προσέγγιση Pig που εφαρμόζει την στρατηγική vertical partitioning.

8.7.1.Ρυθμίσεις για την εκτέλεση των πειραμάτων

Διεξαγάγαμε τα πειράματά μας στο VCL[12] ,ένα βοηθητικό πρόγραμμα υπολογισμού και μια τεχνολογία προσανατολισμένη στην παροχή υπηρεσιών που προσφέρει απομακρυσμένη πρόσβαση σε εικονοποιημένες πηγές(που έχουν γίνει εικονικές).Χρησιμοποιήσαμε clusters όπου κάθε κόμβος είχε τις ελάχιστες προδιαγραφές από μονοπύρηνες ή διπύρηνες μηχανές Intel x86 με 2.33ghz ταχύτητα επεξεργαστή,4gb μνήμη που τρέχουν Red Hat Linux.Χρησιμοποιήσαμε την έκδοση Pig 0.5.0 και Hadoop 0.20.Αυτά τα πειράματα διεξήχθησαν σε ένα cluster με 5 κόμβους με το μέγεθος του μπλοκ να έχει ρυθμιστεί στα 256 Mb.Επίσης καταγράψαμε τεστ κλιμάκωσης σε ένα cluster με 20 κόμβους. Όλα τα αποτελέσματα που καταγράφηκαν ήταν ένας μέσος όρος τριών προσπαθειών.

8.7.2Παραγωγή δεδομένων και εργασίες ελέγχου

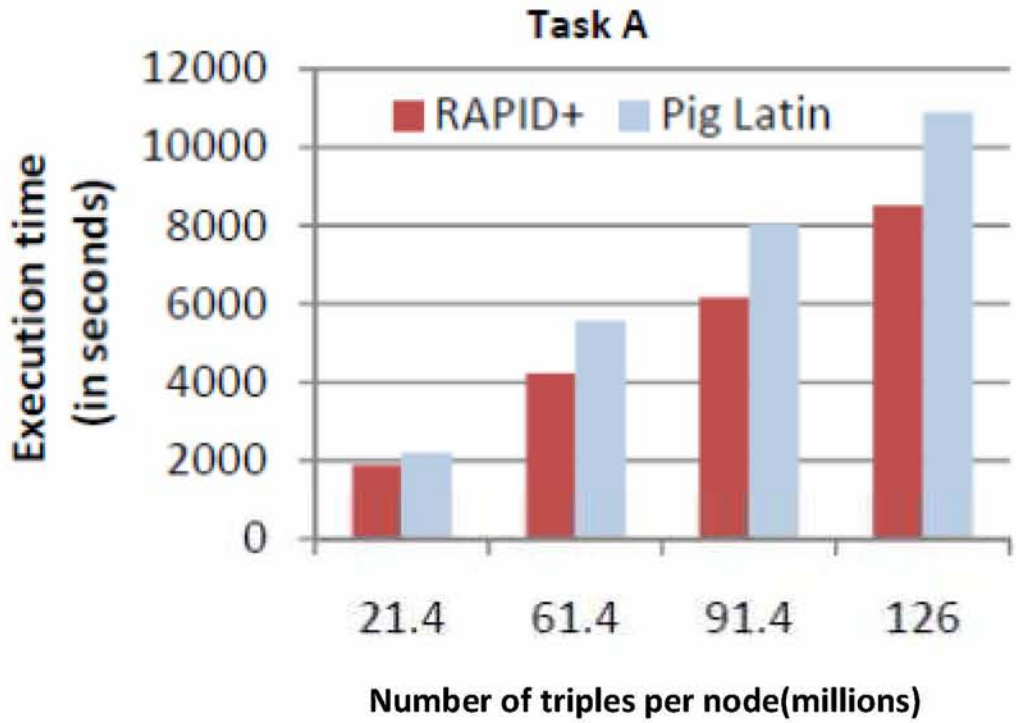
Χρησιμοποιήσαμε ένα συνθετικό σύνολο δεδομένων ,μια τροποποιημένη έκδοση της ανάλυσης των σημείων αναφοράς των δεδομένων που συζητήθηκαν στο {16},το οποίο αρχικά παρήγαγε τα σύνολα δεδομένων rankings και uservisits σαν οριοθετημένα ξεχωριστά αρχεία. Χρησιμοποιήσαμε Perl script για να μετατρέψουμε αυτές τις σχέσεις σε τριαδικές τριπλέτες του τύπου(subject-property-object)όπως φαίνεται στην εικόνα 1β .Κάθε πλειάδα από τη σχέση Ranking μοντελοποιήθηκε σε τέσσερις τριπλέτες ,κάθε μια για τις τρεις στήλες pageURL,pageRank,avgDuration και μια επιπρόσθετη τριπλέτα για να αντιπροσωπεύσει την ιδιότητα type.Παρομοίως κάθε πλειάδα UserVisits μοντελοποιήθηκε σαν ένα σύνολο από δέκα τριπλέτες.Στο κατώτερο άκρο χρησιμοποιήσαμε το σύνολο δεδομένων rankings με περίπου 5,9 εκατομμύρια τριπλέτες ανά κόμβο (193mb/node)και σύνολο δεδομένων userVisits με 15.5εκατομμύρια τριπλέτες ανά κόμβο(500mb/node).Στο υψηλότερο άκρο τεστάραμε με 120 εκατομμύρια τριπλέτες ανά κόμβο (3,9gb/node)της σχέσης userVisits. Αξιολογήσαμε δύο εργασίες, μια που περιλάμβανε βασική αντιστοίχιση προτύπου(taskA)και μια άλλη εργασία που περιλαμβάνει αντιστοίχιση προτύπου ακολουθούμενη από ομαδοποίση/άθροισμα(task B).Ο στόχος της αξιολόγησής μας ήταν η περίπτωση αναφοράς προτύπων με δυο αστέρια και μια σύνδεση μεταξύ των αστεριών.Περιμένουμε ότι με πιο περίπλοκες αναζητήσεις(περισσότερα πρότυπα) η στρατηγική που βασίζεται στην ομαδοποίηση που ταυτόχρονα υπολογίζει τα πρότυπα αστεριού ,θα μας δώσει μεγαλύτερο πλεονέκτημα όσον αφορά την βελτίωση των ποσοστών σε χρόνους εκτέλεσης σε σύγκριση με την προσέγγιση Pig.

8.8 Πειραματικά αποτελέσματα

8.8.1Αναζήτηση αντιστοίχισης προτύπου

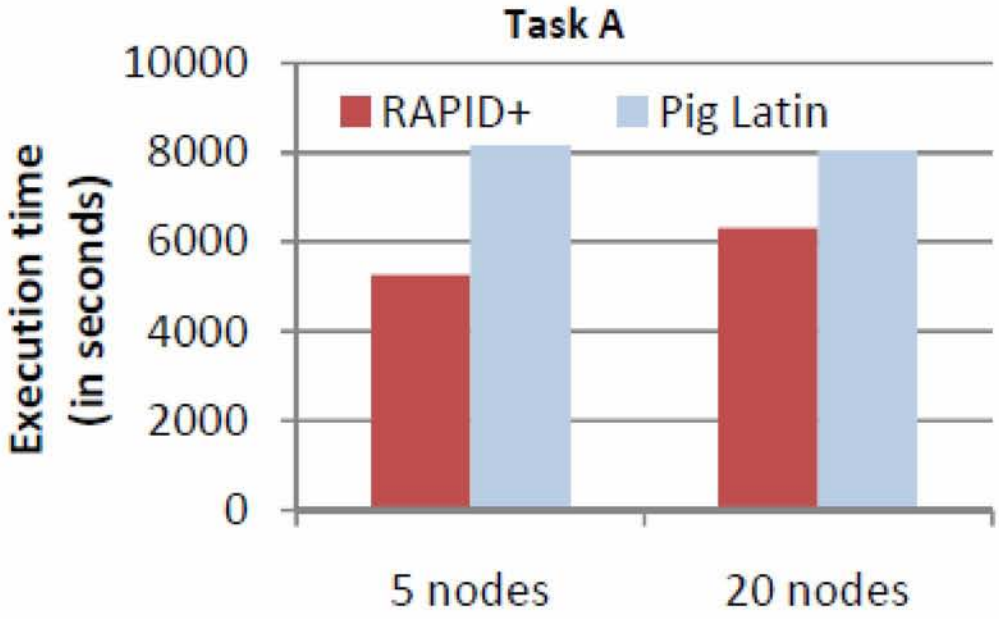
Η εικόνα 6 δείχνει την απόδοση των δυο προσεγγίσεων σε ένα cluster

πέντε κόμβων ρυθμισμένο για την αναζήτηση ταιριάσματος προτύπου. Και για τα τέσσερα μεγέθη δεδομένων βλέπουμε ένα καλό ποσοστό βελτίωσης στους χρόνους εκτέλεσης. Η προσέγγισή μας μεταγλωττίζει σε δύο κύκλους mapreduce (έναν για την φάση ομαδοποίησης στο subject και ένα για την ένωση ανάμεσα στα αστέρια),διαφορετικά από τους τρεις κύκλους mapreduce (καθένας για τις δυο αστεροειδής ενώσεις και μια ένωση)στην προσέγγιση Pig,επομένως μας δίνει οικονομία χρόνου με μόνο δυο πρότυπα αστεριού. Πιο αναλυτικές αναζητήσεις σε δεδομένα RDF που περιλαμβάνουν υπολογισμούς διαφόρων προτύπων θα επωφεληθούν από αυτήν την προσέγγιση. Επίσης επεκτείναμε το πείραμα σε ένα cluster είκοσι κόμβων με περίπου 1110 εκατομμύρια τριπλέτες (36Gb)των userVisits και 379εκατομμύρια τριπλέτες (12,7GB)συνόλων δεδομένων rankings,με ένα σύνολο από 1490 εκατομμύρια τριπλέτες (48,7 GB).Η εικόνα 7 δείχνει το αποτέλεσμα για την μελέτη κλιμάκωσης και τις βελτιώσεις επίδοσης που επετεύχθησαν χρησιμοποιώντας τους τελεστές RAPID+.



εικόνα 6:κόστος ανάλυσης για την εργασία A(cluster 5 κόμβων)

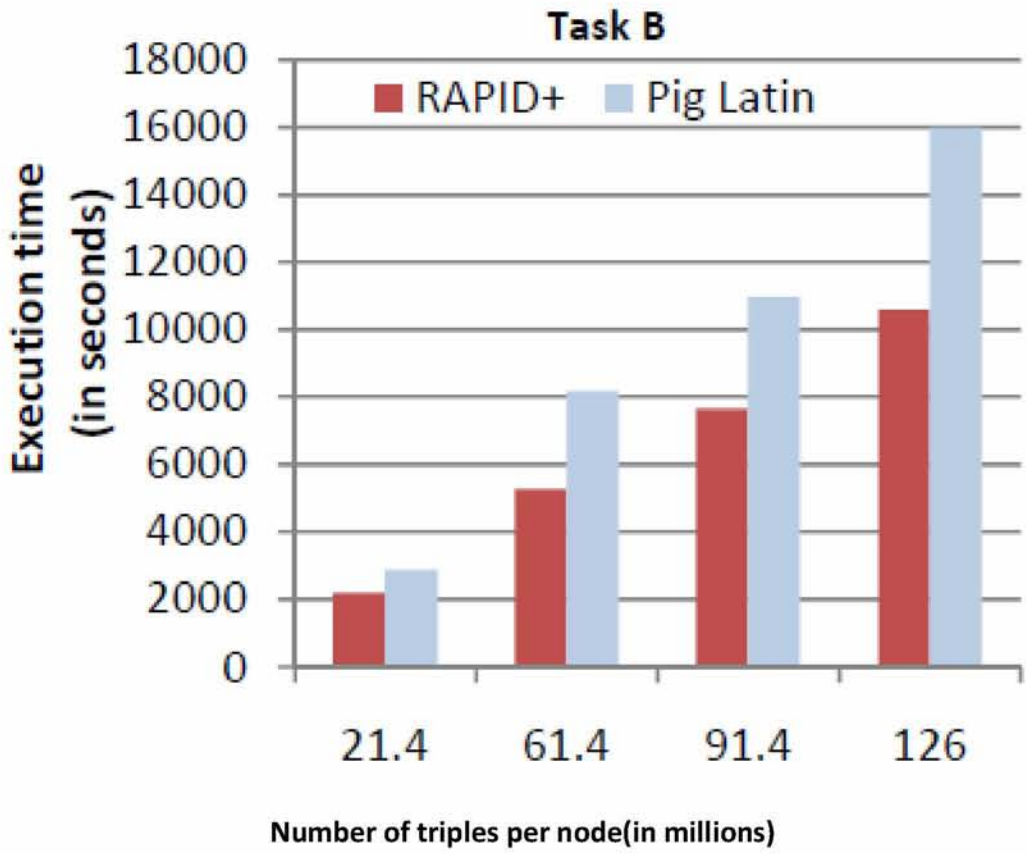
Πηγή:P.Ravindra,V.Deshpande,K.Anyanwu,Towards Scalable RDF Graph Analytics on MapReduce



εικόνα 7:Κλιμάκωση της μελέτης για την εργασία A(5 κόμβοι vs 20 κόμβοι) Πηγή:P.Ravindra,V.Deshpande,K.Anyanwu,Towards Scalable RDF Graph Analytics on MapReduce

8.8.2Αντιστοίχιση προτύπου με ομαδοποίηση/άθροισμα

Η εικόνα 8 δείχνει την μείωση κόστους που έχουμε εκτελώντας τα δυο lookaheads. Πρώτον κατά την διάρκεια της ομαδοποίησης-φιλτραρίσματος επανασυναρμολογούμε τους σάκους για να προετοιμαστούμε για την ένωση μεταξύ των αστεριών και δεύτερον κατά την διάρκεια της φάσης σύνδεσης προετοιμάζουμε τους σάκους για την φάση ομαδοποίησης. Εδώ πάλι η αρχική εξοικονόμηση χρόνου επέρχεται από τους υπολογισμούς των προτύπων αστεριού. Επίσης μειώνουμε κάποια κόστη εισόδου-εξόδου ελαχιστοποιώντας τον αριθμό των πλειάδων που δεν χρειάζονται πια π.χ. μετά από κάθε φάση φιλτραρίσματος, ένωσης και ομαδοποίησης. Βλέπουμε σχεδόν μόνιμο ποσοστό βελτίωσης στον χρόνο εκτέλεσης σε όλα τα μεγέθη δεδομένων. Σημειώστε ότι τώρα δεν εφαρμόζουμε την τυπική συνάρτηση Pig για να υπολογίσουμε τα επί μέρους αθροίσματα τα οποία θα προστεθούν στην εξοικονόμηση μας.



εικόνα 8:Κόστος της ανάλυσης για την εργασία B(cluster 5 κόμβων)

Πηγή:P.Ravindra,V.Deshpande,K.Anyanwu,Towards Scalable RDF Graph Analytics on MapReduce

8.9 Συμπέρασμα

Παρουσιάσαμε μια προσέγγιση για να επιτύχουμε καλύτερη παραλληλοποίηση στην επεξεργασία αναλυτικών αναζητήσεων σε μοντέλα γράφων RDF.Επεκτείναμε την βιβλιοθήκη του Pig Latin για να συμπεριλάβουμε συναρτήσεις που βοηθούν στο operator coalescing και looahead επεξεργασία για να μειώσουμε τα κόστη εισόδου -εξόδου που προκύπτουν από την επαναλαμβανόμενη επεξεργασία και υλοποίηση των ενδιάμεσων αποτελεσμάτων. Η μελέτη δείχνει υποσχόμενες βελτιώσεις στον χρόνο εκτέλεσης και για την αντιστοίχιση προτύπων και για τις απλές αναλυτικές αναζητήσεις .

Βιβλιογραφία

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," In: Proc. Of OSDI, 2004.
[2] C. Olston, B. Reed, U. Srivastava, R. Kumar and A. Tomkins, "Pig Latin: a not-so-foreign language for data processing," ACM SIGMOD, 2008.
[3] D.J. Abadi, A. Marcus, S.R. Madden and K. Hollenbach, "Scalable Semantic Web Data Management Using Vertical Partitioning,", VLDB, 2007.
[4] H. Yang, A. Dasdan, R. Hsiao, Jr., D.S. Parker, "Map-reducemerge: simplified relational data processing on large clusters," SIGMOD, 2007.
[5] R. Chaiken, B. Jenkins, P. Larson, B. Ramsey, D. Shakib, S. Weaver and J. Zhou, "SCOPE: easy and efficient parallel processing of massive data sets," PVLDB, 2008.
[6] Yu,Y., Isard, M., Fetterly,D., Badiu,M ., Erlingsson,U., Gunda,P.K. , and Currey,J.: DryadLINQ: A system for generalpurpose distributed data-parallel computing using a high-level language. OSDI 2008
[7] R. Pike, S. Dorward, R. Griesemer and S. Quinlan, "Interpreting the data: Parallel analysis with Sawzall," Scientific Programming, 2005.
[8] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi and A. Silberschatz, "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads," VLDB, 2009.
[9] R. Sridhar, P. Ravindra and K. Anyanwu, "RAPID: Enabling scalable ad-hoc analytics on the semantic web," ISWC, 2009.
[10] E. Prud'hommeaux and A. Seaborne, "SPARQL query language for RDF," Technical report, World Wide Web Consortium, 2005, http://www.w3.org/TR/rdf-sparql-query.
[11] Apache Projects Proceedings : http://hadoop.apache.org/core/
[12] VCL Setup at NC State University : https://vcl.ncsu.edu/
[13] JAQL : http://code.google.com/p/jaql
P.Ravindra, V.Deshpande and K.Anyanwu, "Towards Scalable RDF Graph Analytics on MapReduce,"ACM, 2010.

Παράρτημα

Κώδικας WordCount

```
 1 package org.myorg;
 2
 3 import java.io.IOException;
 4 import java.util.*;
 5
 6 import org.apache.hadoop.fs.Path;
 7 import org.apache.hadoop.conf.*;
 8 import org.apache.hadoop.io.*;
 9 import org.apache.hadoop.mapreduce.*;
10 import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
11 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
13 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
14
15 public class WordCount {
16
17  public static class Map extends Mapper<LongWritable, Text, Text,
IntWritable> {
18     private final static IntWritable one = new IntWritable(1);
19     private Text word = new Text();
20
21     public void map(LongWritable key, Text value, Context context)
throws IOException, InterruptedException {
22         String line = value.toString();
23         StringTokenizer tokenizer = new StringTokenizer(line);
24         while (tokenizer.hasMoreTokens()) {
25             word.set(tokenizer.nextToken());
26             context.write(word, one);
27         }
28     }
29  }
30
31  public static class Reduce extends Reducer<Text, IntWritable,
Text, IntWritable> {
32
33     public void reduce(Text key, Iterable<IntWritable> values,
Context context)
34       throws IOException, InterruptedException {
35         int sum = 0;
36         for (IntWritable val : values) {
37             sum += val.get();
38         }
39         context.write(key, new IntWritable(sum));
40     }
41  }
42
43  public static void main(String[] args) throws Exception {
44     Configuration conf = new Configuration();
45
46         Job job = new Job(conf, "wordcount");
47
```

```
48      job.setOutputKeyClass(Text.class);
49      job.setOutputValueClass(IntWritable.class);
50
51      job.setMapperClass(Map.class);
52      job.setReducerClass(Reduce.class);
53
54      job.setInputFormatClass(TextInputFormat.class);
55      job.setOutputFormatClass(TextOutputFormat.class);
56
57      FileInputFormat.addInputPath(job, new Path(args[0]));
58      FileOutputFormat.setOutputPath(job, new Path(args[1]));
59
60      job.waitForCompletion(true);
61   }
62
63 }
```

Βιβλιογραφία

- Front Page – Hadoop Wiki: http://wiki.apache.org/hadoop/
- Apache Hadoop Wikipedia: http://en.wikipedia.org/wiki/Hadoop
- MapReduce Wikipedia: http://en.wikipedia.org/wiki/MapReduce
- Welcome to Apache Hadoop: http://hadoop.apache.org
- Google Research Publication: The Google File System http://labs.google.com/papers/gfs.html
- Google Research Publication: MapReduce http://labs.google.com/papers/mapreduce.html
- Tom White, Hadoop: The Difinitive Guide, California: O'Reilly, 2009
- Jason Venner, Pro Hadoop, New York: Springer – Verlag, 2009
- Chuck Lam, Hadoop In Action, Connecticut: Manning, 2011
- J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters"
- Welcome to Apache Hadoop: http://hadoop.apache.org
- Getting Started with Hadoop http://wiki.apache.org/hadoop/GettingStartedWithHadoop
- Project Description http://wiki.apache.org/hadoop/ProjectDescription
- How to debug MapReduce programs http://wiki.apache.org/hadoop/HowToDebugMapReducePrograms
- Hadoop API Overview http://hadoop.apache.org/common/docs/current/api/overview-summary.html
- Running Hadoop On Ubuntu Linux(Single-Node Cluster), Michael G. Noll, http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/
- The Outline of Science, Vol. 1 (of 4) by J. Arthur Thomson : http://www.gutenberg.org/ebooks/20417
- The Notebooks of Leonardo Da Vinci : http://www.gutenberg.org/ebooks/5000
- Ulysses by James Joyce : http://www.gutenberg.org/ebooks/4300
- A. Berger and J.Lafferty, "Information retrieval as statistical translation," In SIGIR 1999.
- P. F. Brown, V. J. Pietra, S. A. D. Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," Computational Linguistics, 1993.
- K. Jarvelin and J. Kekalainen, "Cumulated gain-based evaluation of ir techniques," ACM Trans. Inf. Syst., 2002.
- M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks: estimating the click-through rate for new ads," In WWW, 2007.
- B. Shaparenko, O. Cetin, and R. Iyer, "Data driven text features for sponsored search click prediction," In AdKDD Workshop, 2009.
- D. Hillard, S. Schroedl, E. Manavoglu, H. Raghavan, C. Leggeter, "Improving ad relevance in Sponsored Search," New York 2010, www.atbrox.com/about

- D. Carmel, H. Roitman, and N. Zwerdling. Enhancing cluster labeling using wikipedia. In SIGIR '09, pages 139–146, New York, NY, USA, 2009. ACM.
- D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? In SIGIR '06, pages 390–397. ACM Press, 2006.
- L. Chen and K. Sycara. Webmate: a personal agent for browsing and searching. In AGENTS '98, New York, NY, USA, 1998. ACM.
- S. Kullback and R. A. Leibler. On information and sufficiency. The Annals of Mathematical Statistics, 22(1):79–86, 1951
- C. D. Manning, P. Raghavan, and H. Schutze. Introduction to Information Retrieval. Cambridge University Press, 2008.
- K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In WWW, pages 675–684, 2004
- M. Shmueli-Scheuer, H. Roitman, D. Carmel, Y. Mass, D. Konopnicki, "Extracting User Profile from Large Scale Data", 2010.
- R. Belohlavek and V. Vychodil, "Discovery of optimal factors in binary data via a novel method of matrix decomposition," Journal of Computer and System Sciences(to appear).
- A. Berry, J.-P Bordat, A. Sigayret, "A local approach to concept generation. Annals of Mathematics and Artificial Intelligence," 2007.
- C. Carpineto and G. Romano, "Concept data analysis: Theory and applications," J. Wiley, 2004.
- J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large cluster," Commun. ACM, 2008.
- H. Fu and E.M. Nguifo, "A parallel algorithm to generate formal concepts for large Data," In: Eklund, P. (ed.) ICFCA 2004. LNCS (LNAI), vol. 2961, pp. 394–401, Springer, Heidelberg (2004)
- B. Ganter, "Two basic algorithms in concept analysis (Technical Report FB4-Preprint No. 831)," TH Darmstadt, 1984.
- B. Ganter and R. Wille, "Formal concept analysis," Mathematical foundations, Springer, Berlin 1999.
- Hadoop Core Framework: http://hadoop.apache.org/
- S. Hettich and S.D. Bay, "The UCI KDD Archive University of California, Irvine, School of Information and Computer Sciences," 1999.
- J.F.D. Kengue, P. Valtchev and C.T. Djamegni, "A parallel algorithm for lattice construction," In: Ganter, B., Godin, R. (eds.) ICFCA 2005. LNCS (LNAI), vol. 3403, pp. 249–264. Springer, Heidelberg 2005.
- P. Miettinen, T. Mielik¨ainen, A. Gionis, G. Das and H. Mannila, "The discrete basis Problem," In: F¨urnkranz, J., Scheffer, T., Spiliopoulou, M. (eds.) PKDD 2006. LNCS (LNAI), vol. 4213, pp. 335–346. Springer, Heidelberg, 2006.
- P. Krajca, J. Outrata and V. Vychodil, "Parallel Recursive Algorithm for FCA," In: Belohlavek, R., Kuznetsov, S.O. (eds.) Proc. CLA 2008, vol. 433, pp. 71–82. CEUR WS, 2008, ISBN 978–80–244–2111–7.

- P. Krajca, J. Outrata and V. Vychodil, "Parallel Algorithm for Computing Fixpoints of Galois Connections," Annals of Mathematics and Artificial Intelligence (submitted)
- S. Kuznetsov, "Interpretation on graphs and complexity characteristics of a search for specific patterns," Automatic Documentation and Mathematical Linguistics 24(1), 37–45, 1989.
- S. Kuznetsov, "A fast algorithm for computing all intersections of objects in a finite semi-lattice (Bystry_i algoritm postroeni_ vseh pereseqeni_i ob_ektov iz koneqno_i polurexetki, in Russian)," Automatic Documentation and Mathematical Linguistics 27(5), 11–21, 1993.
- S.O. Kuznetsov, "Learning of simple conceptual graphs from positive and negative Examples," In: ´Zytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 384–391. Springer, Heidelberg, 1999.
- S. Kuznetsov and S. Obiedkov, "Comparing performance of algorithms for generating concept lattices," J. Exp. Theor. Artif. Int. 14, 189–216, 2002.
- C. Lindig, "Fast concept analysis," In: Working with Conceptual Structures - Contributions to ICCS 2000, pp. 152–161, Shaker Verlag, Aachen, 2000.
- P. Krajca and V.Vychodil, "Distributed Algorithm for Computing Formal Concepts Using Map-Reduce Framework," Springer – Verlag Berlin, 2009.
- Mapreduce & Hadoop Algorithms in Academic Papers (4th update – May 2011): www.atbrox.com/about
- M. Collins and Y. Singer, "Unsupervised Models for Named Entity Classification," , In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-99), 1998.
- M. A. Hearst, "Automatic Acquisition of Hyponyms from Large Text Corpora," In Proceedings of Conference on Computational Linguistics (COLING-92), Nantes, France, 1992.
- T. Hasegawa, S. Sekine, R. Grishman, "Discovering Relations among Named Entities from Large Corpora," In the proceedings of Association of Computational Linguistics (ACL-04). Barcelona, Spain, 2004.
- Moffat and J. Zobel, "Inverted Files for Text Search Engines," ACM Computing Surveys, 38(2):1-56, July 2006.
- P. Resnik and A. Elkiss, "The Linguist's Search Engine: Getting Started Guide," Technical Report: LAMP-TR-108/CS-TR-4541/UMIACS-TR-2003-109, University of Maryland, College Park, 2003.
- S. Sekine, "A Linguistic Knowledge Discovery Tool," In Proceeding of COLING08. Stanford tagger: http://nlp.stanford.edu/software/tagger.shtml, 2008
- OAK system: http://nlp.cs.nyu.edu/oak
- Wikipedia tagged data: http://nlp.cs.nyu.edu/wikipedia-data
- S.Sekine and K.Dalwani, "Ngram Search Engine with Patterns Combining Token,POS,Chunk and NE Information
- www.wikipedia.org
- J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," In: Proc. Of OSDI, 2004.
- C. Olston, B. Reed, U. Srivastava, R. Kumar and A. Tomkins, "Pig Latin: a not-so-foreign language for data processing," ACM SIGMOD, 2008.
- D.J. Abadi, A. Marcus, S.R. Madden and K. Hollenbach, "Scalable Semantic Web

Data Management Using Vertical Partitioning,", VLDB, 2007.
- H. Yang, A. Dasdan, R. Hsiao, Jr., D.S. Parker, "Map-reducemerge: simplified relational data processing on large clusters," SIGMOD, 2007.
- R. Chaiken, B. Jenkins, P. Larson, B. Ramsey, D. Shakib, S. Weaver and J. Zhou, "SCOPE: easy and efficient parallel processing of massive data sets," PVLDB, 2008.
- Yu,Y., Isard, M., Fetterly,D., Badiu,M ., Erlingsson,U., Gunda,P.K. , and Currey,J.: DryadLINQ: A system for generalpurpose distributed data-parallel computing using a high-level language. OSDI 2008
- R. Pike, S. Dorward, R. Griesemer and S. Quinlan, "Interpreting the data: Parallel analysis with Sawzall," Scientific Programming, 2005.
- A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi and A. Silberschatz, "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads," VLDB, 2009.
- R. Sridhar, P. Ravindra and K. Anyanwu, "RAPID: Enabling scalable ad-hoc analytics on the semantic web," ISWC, 2009.
- E. Prud'hommeaux and A. Seaborne, "SPARQL query language for RDF," Technical report, World Wide Web Consortium, 2005, http://www.w3.org/TR/rdf-sparql-query.
- Apache Projects Proceedings : http://hadoop.apache.org/core/
- VCL Setup at NC State University : https://vcl.ncsu.edu/
- JAQL : http://code.google.com/p/jaql
- P.Ravindra, V.Deshpande and K.Anyanwu, "Towards Scalable RDF Graph Analytics on MapReduce,"ACM, 2010.