ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Ακ. έτος 2013-2014, 9ο Εξάμηνο ΗΜ&ΜΥ

Ν. Κοζύρης

Εξαμηνιαία Εργασία

Εισαγωγή στο MapReduce και στις βάσεις NoSQL

1 Εισαγωγή

Σε αυτή την άσκηση θα χρησιμοποιήσουμε αλγόριθμους MapReduce για να κάνουμε αποθήκευση και ανάλυση σε έναν αριθμό από σετ δεδομένων.

Στο πρώτο μέρος (Ενότητα 2) θα χρησιμοποιήσουμε το Hadoop Distributed File System για την αποθήκευση των δεδομένων και το μοντέλο MapReduce για την επεξεργασία των 2 αυτών σετ δεδομένων.

Στο δεύτερο μέρος (Ενότητα 3) της άσκησης θα χρησιμοποιήσουμε την HBase., μια κατανεμημένη βάση NoSQL για να κάνουμε μαζική εισαγωγή και επεξεργασία των προηγούμενων δεδομένων στην βάση με σκοπό την δυνατότητα εκτέλεσης ερωτημάτων.

Στο τρίτο μέρος (Ενότητα) της άσκησης θα μελετήσουμε τις δυνατότητες κλιμάκωσης που προσφέρει το κατανεμημένο μοντέλο επεξεργασίας MapReduce. Θα εκτελέσουμε ένα υποσύνολο των προηγούμενων ερωτημάτων χρησιμοποιώντας μεγαλύτερα σετ δεδομένων σε συστοιχίες Hadoop διαφορετικών μεγεθών στον okeanos.

1.1 Σετ Δεδομένων (Datasets) που θα χρησιμοποιηθούν

Θα χρησιμοποιήσουμε ένα σετ από πραγματικά ερωτήματα χρηστών μηχανών αναζήτησης και ένα σετ με τους τίτλους όλων των άρθρων της Wikipedia.

Για ερωτήματα χρηστών θα χρησιμοποιήσουμε το dataset της America on Line[1] (AOL) που αποτελείται από 20 εκατομμύρια ερωτήματα που έγιναν από 650,000 χρήστες σε διάστημα 3 μηνών. Το αρχείο περιέχει tab-delimited γραμμές της μορφής

```
1038 max bretos 2006-03-09 22:37:11 1 http://www.soccerloop.com
```

Όπου το πρώτο πεδίο αποτελεί το userid του χρήστη, το δεύτερο πεδίο τα keywords του ερωτήματος διαχωρισμένα με κενά, το τρίτο πεδίο την ημερομηνία του ερωτήματος, το τέταρτο πεδίο την θέση του result και το τελευταίο πεδίο το result που έκανε κλικ ο

[1] http://en.wikipedia.org/wiki/AOL_search_data_leak

χρήστης. Σε περίπτωση που ο χρήστης δεν επέλεξε κάποιο result ο χρήστης, το τέταρτο και πέμπτο πεδίο είναι κενά. Το AOL dataset της εργασίας μπορείτε να το κατεβάσετε από εδώ:

www.cslab.ntua.gr/~ikons/user-ct-test-collection-01.txt.gz

Όσον αφορά το dataset των τίτλων της Wikipedia, κάθε γραμμή του αρχείου περιέχει και από έναν τίτλο. Οι λέξεις του τίτλου είναι διαχωρισμένες με το σύμβολο "_". Το αρχείο που περιέχει όλους τους τίτλους των άρθρων της Wikipedia μπορείτε να το κατεβάσετε από εδώ:

http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-all-titles-in-ns0.gz

Κάθε γραμμή του αρχείου περιέχει και από έναν τίτλο. Οι λέξεις του τίτλου είναι διαχωρισμένες με το σύμβολο "_"

Τα παραπάνω datasets πρέπει να ανέβουν στο hdfs.

Κατά την φάση της ανάλυσης, για να μειώσουμε τον θόρυβο των αποτελεσμάτων θα χρειαστεί να χρησιμοποιήσουμε την παρακάτω stop-list με keywords:

http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

Η λίστα αυτή περιέχει πολύ συχνές λέξεις που υπάρχουν στην αγγλική γλώσσα. Κατά την ανάγνωση των dataset/queryset θα πρέπει να αποκλείουμε λέξεις που υπάρχουν στην παραπάνω stoplist.

Επίσης, κατά την εγκατάσταση του hadoop χρειάζεται να αλλάξουμε το default block size του hdfs. Αυτό γίνεται βάζοντας το παρακάτω entry στο αρχείο hdfs-site.xml

```
<property>
      <name>dfs.block.size</name>
      <value>33554432</value>
</property>
```

Ο λόγος που το κάνουμε αυτό είναι επειδή το hadoop εκτελεί έναν mapper ανά hdfs block. Επειδή τα dataset είναι σχετικά μικρά (περίπου 200MB το καθένα) το default block size θα τα σπάσει σε λίγα κομμάτια, και δεν θα εκτελεστούν αρκετοί mappers για να δούμε τον παραλληλισμό των εργασιών.

Βασικό είναι να βλέπουμε το Hadoop API που περιέχει όλες τις μεθόδους/κλάσεις του hadoop. Αυτό βρίσκεται σε αυτή την διεύθυνση:

http://hadoop.apache.org/common/docs/current/api/index.html?overview-summary.html

και στον κατάλογο docs/api/index.html του hadoop-1.0.4.tar.gz που έχουμε κατεβάσει.

Δεν χρειάζεται να κατεβάσουμε/αποσυμπιέσουμε τα datasets τοπικά στο δικό μας μηχάνημα. Μπορούμε να τα κατεβάσουμε/αποσυμπιέσουμε/ανεβάσουμε κατευθείαν στο

hdfs με linux piping[2] από τον server του okeanos. Οι εντολές για να το κάνουμε αυτό είναι οι wget, tar και hadoop fs

Δημιουργούμε υποσύνολα των datasets για την ανάπτυξη, για να τελειώνουν πιο γρήγορα τα map/reduce jobs. Πχ, μπορούμε να δημιουργήσουμε ένα enwiki_titles_small.txt με 1000 τίτλους και ένα aol_queries.txt με 1000 queries, έτσι ώστε να δοκιμάζουμε τα προγράμματά μας πριν τους δώσουμε τα μεγάλα datasets.

2 Εισαγωγή στο MapReduce

Το AOL dataset αυτού του μέρους της εργασίας μπορείτε να το κατεβάσετε από εδώ:

www.cslab.ntua.gr/~ikons/user-ct-test-collection-01.txt.gz

Το αρχείο που περιέχει όλους τους τίτλους των άρθρων της Wikipedia μπορείτε να το κατεβάσετε από εδώ:

http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-all-titles-in-ns0.gz

2.1 Υπολογισμός αριθμού αναζητήσεων ανά ημέρα.

Σε αυτή την φάση, ζητείται ένα MapReduce πρόγραμμα που θα υπολογίζει τις τιμές για την εξαγωγή ενός γραφήματος που δείχνει για κάθε διαφορετική μέρα της συνολικής χρονικής περιόδου, τον αριθμό των αναζητήσεων που έγιναν για την μέρα αυτή. Το γράφημα αυτό θα έχει στον άξονα Χ ταξινομημένες τις ημερομηνίες (πχ 2006-03-09, 2006-03-10, κλπ) και στον άξονα Υ τον αριθμό των queries που έγιναν για κάθε μέρα.

2.2 Υπολογισμός ποσοστού «επιτυχών» και «ανεπιτυχών» αναζητήσεων.

Ως επιτυχείς αναζητήσεις θεωρούμε τις αναζητήσεις όπου οι χρήστες επέλεξαν να επισκεφτούν μια σελίδα από τα αποτελέσματα. Αντίστοιχα, ανεπιτυχείς είναι οι αναζητήσεις όπου οι χρήστες δεν επέλεξαν να επισκεφτούν κάποιο αποτέλεσμα.

Ζητείται ένα πρόγραμμα mapreduce που θα εξάγει τα ποσοστά των επιτυχών και των ανεπιτυχών αναζητήσεων.

2.3 Λίστα ιστοσελίδων που επισκέφτηκαν παραπάνω από 10 διαφορετικοί χρήστες.

Εδώ ζητείται η εξαγωγή μιας λίστας που θα περιέχει δύο στήλες: Η πρώτη στήλη θα περιέχει το url των ιστοσελίδων που επισκέφτηκαν πάνω από 10 διαφορετικοί χρήστες, και η δεύτερη στήλη τον αριθμό των χρηστών που επισκέφτηκαν την κάθε ιστοσελίδα.

2.4 Εύρεση δημοφιλών λέξεων κλειδιών των ερωτημάτων της AOL.

Σε αυτή την φάση θα υπολογίσουμε την συχνότητα εμφάνισης των λέξεων κλειδιών στα ερωτήματα του America On Line Dataset. Ζητούμενο είναι η κατασκευή ενός map/reduce

[2] http://wiki.linuxquestions.org/wiki/Piping

προγράμματος που θα παίρνει σαν input το αρχείο των aol queries και θα παράγει ένα αρχείο της παρακάτω μορφής:

```
VeryPopularKeyword      124324
PopularKeyword           34053
LessPopularKeyword        2345
UnpopularKeyword            35
```

Το αρχείο θα περιέχει όλα τα διαφορετικά unique keywords που υπάρχουν στο dataset μαζί με τον αριθμό εμφανίσεων του καθενός. Η ταξινόμηση θα είναι όπως στον παραπάνω πίνακα: το πιο δημοφιλές keyword στην αρχή του αρχείου. Στην αναφορά ζητείται η λίστα με τα 50 πιο δημοφιλή keywords (με την αφαίρεση "άσεμνων" λέξεων) καθώς και ο συνολικός αριθμός των unique keywords που μετρήσατε.

Το αρχείο με την κατανομή των ερωτημάτων θα χρησιμοποιηθεί στην άσκηση 2 σαν γεννήτρια για την παραγωγή συνθετικής "κίνησης".

Hint3: Μπορείτε να το κάνετε σε 2 map/reduce jobs: το πρώτο θα βγάλει τα keywords με την σειρά εμφανίσεως του καθενός και το δεύτερο map/reduce θα τα ταξινομήσει με σειρά εμφάνισης. Σημαντική εντολή για να κάνετε την αντίστροφη ταξινόμηση είναι η `setOutputKeyComparatorClass(LongWritable.DecreasingComparator.class);` έτσι ώστε να εξαχθούν οι λέξεις με τις περισσότερες εμφανίσεις στην αρχή. Το δεύτερο map/reduce μπορείτε να βάλετε έναν reducer για να βγει ένα ενιαίο τελικό αρχείο.

Μετά την εξαγωγή της ταξινομημένης λίστας, ζητείται η σχεδίαση ενός διαγράμματος τιμών <x,y> όπου στον άξονα των x θα υπάρχουν τα keywords ταξινομημένα σύμφωνα με την παραπάνω λίστα και στον άξονα των y ο αριθμός εμφανίσεων του κάθε keyword. Τι παρατηρείτε?

Hint4: Δείτε το άρθρο της Wikipedia για το zipf's law[3]

2.5 Υπολογισμός ιστογράμματος της λεξικογραφικής κατανομής των λέξεων κλειδιών των άρθρων των τίτλων της wikipedia.

Όπως είδαμε στο προηγούμενο ερώτημα, η κατανομή των δεδομένων δεν είναι πάντα ομοιόμορφη και γνωστή εκ των προτέρων. Αυτό συμβαίνει αρκετά συχνά και σε δεδομένα κειμένου. Οι ανομοιόμορφες κατανομές δημιουργούν προβλήματα σε εφαρμογές που πρέπει να επεξεργαστούν τέτοιου είδους δεδομένα αφού πρώτα τα σπάσουν σε "κομμάτια", όπως πχ κάνει το MapReduce: υπάρχει περίπτωση ένα κομμάτι να δημιουργήσει περισσότερη εργασία από ότι τα άλλα κομμάτια, καθυστερώντας το task που έχει αναλάβει την επεξεργασία του και κατά συνέπεια όλο τον υπολογισμό.

Πάρτε για παράδειγμα το πρόγραμμα WordCount του hadoop που μετράει τον αριθμό εμφανίσεων των λέξεων σε ένα σύνολο κειμένων: Η μέτρηση γίνεται με την αποστολή λέξεων σε έναν αριθμό από reducers οι οποίοι μετράνε τον αριθμό εμφανίσεων των λέξεων. Με ποιον τρόπο όμως αποφασίζεται ποιος reducer θα πάρει ποιες συγκεκριμένες λέξεις? Ένας τρόπος θα ήταν να χωρίζαμε τον λεξικογραφικό χώρο [A…Z] σε "ίσια" κομμάτια, όπου ο πρώτος reducer θα έπαιρνε πχ τις λέξεις που ξεκινάνε από [A..C], ο

[3] http://en.wikipedia.org/wiki/Zipf's_law

δεύτερος τις λέξεις από [D…F], κλπ. Σε περιπτώσεις όμως άνισης κατανομής, αυτό δεν είναι δίκαιο, καθώς ένας reducer μπορεί να πάρει λιγότερες λέξεις από τους άλλους. Πχ ο τελευταίος reducer που θα έπαιρνε τις λέξεις [X…Z] θα είχε λιγότερα αντικείμενα σε σχέση με τους άλλους, καθώς δεν υπάρχουν πολλές λέξεις που ξεκινούν από X,Y,Z.

Για τον ισομερή καταμερισμό δεδομένων σε περιπτώσεις που δεν είναι γνωστή εκ των προτέρων η κατανομή που ακολουθούν μπορούμε να χρησιμοποιήσουμε τα ιστογράμματα[4]. Τα ιστογράμματα χρησιμοποιούνται από τις βάσεις δεδομένων όπως SQL server, MySQL, κλπ. Ένα ιστόγραμμα *προσεγγίζει* την ακριβή κατανομή των τιμών των δεδομένων στον χώρο. Ένα ιστόγραμμα πάνω σε ένα σύνολο δεδομένων κατασκευάζεται διαχωρίζοντας την κατανομή δεδομένων σε β >=1 ξένα μεταξύ τους υποσύνολα (τα ονομάζουμε κάδους/bins) και προσεγγίζοντας τις συχνότητες και τις τιμές μέσα σε κάθε κάδο με κάποιο κοινό τρόπο.

2.5.1 Πρώτο μέρος

Ζητείται η κατασκευή ενός MapReduce προγράμματος που θα υπολογίζει την λεξικογραφική κατανομή που ακολουθούν οι λέξεις κλειδιά των τίτλων άρθρων της wikipedia με διαφορετικό βαθμό ακρίβειας. Για το ιστόγραμμα θεωρούμε 28 υποσύνολα (buckets): ένα για κάθε γράμμα της αγγλικής αλφαβήτου, ένα υποσύνολο για τους αριθμούς [0…9] και έναν για τα σύμβολα ~!@#$%^&*()_+{}|:"<>?[]\;',./.

Το πρώτο MapReduce πρόγραμμα που θα κατασκευάσετε θα υπολογίσει την **ακριβή** κατανομή των λέξεων κλειδιών των τίτλων (histogram_full). Ζητείται ένας πίνακας της μορφής:

Εύρος τιμών	**Ποσοστό εμφανίσεων**
~!@#$%^&*()_+{}|:"<>?[]\;',./	X_1%
[0…9]	X_2%
A*	X_3%
B*	X_4%
….	….
Y*	X_{27}%
Z*	X_{28}%

Το δεύτερο mapReduce θα υπολογίζει με **προσέγγιση** την κατανομή των λέξεων κλειδιών των τίτλων (ιστόγραμμα). Η προσέγγιση γίνεται κατά την φάση map: αντί να αφήσουμε τους mappers να επεξεργαστούν όλα τα δεδομένα, τους τερματίζουμε μόλις έχουν επεξεργαστεί ένα x αριθμό από εγγραφές. Όσο το x μεγαλώνει, τόσο μεγαλύτερη προσέγγιση έχουμε.
Θα χρησιμοποιήσετε 2 βαθμούς προσέγγισης: Την πρώτη φορά θα τερματίζετε τους mappers μετά από 50 επαναλήψεις (histogram_50) και την δεύτερη μετά από 1000

[4] http://el.wikipedia.org/wiki/%CE%99%CF%83%CF%84%CF%8C%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1

επαναλήψεις (histogram_1000). Ζητείται επίσης η σχεδίαση των ιστογραμμάτων histogram_full, histogram_50 και histogram_1000 στους ίδιους άξονες.

2.5.2 Δεύτερο μέρος

Το επόμενο βήμα της άσκησης είναι να δημιουργήσετε ένα πρόγραμμα MapReduce το οποίο θα εκτελεί λεξικογραφική ταξινόμηση στις λέξεις κλειδιά των τίτλων των άρθρων της wikipedia. Το τελικό hdfs output του προγράμματος θα είναι αρχεία της μορφής:

File1

```
aaaaa
aaaab
aacaa
baaahhg
...
ffasaf
```

File2

```
ffasi
jsdfgs
...
lasdf
```

……

FileN

```
yasadfg
...
zxcxzcza
```

όπου N ο αριθμός των reducers που θα επιλέξετε. Για την άσκηση επιλέξτε 10 reducers. Τα αρχεία θα περιέχουν ταξινομημένα τα keywords, τόσο μέσα σε κάθε αρχείο (η επόμενη γραμμή θα περιέχει το επόμενο λεξικογραφικά keyword) όσο και μεταξύ διαφορετικών αρχείων (το αρχείο File2 θα περιέχει επόμενα λεξικογραφικά keywords από το File1).
Για να το κάνετε αυτό θα χρησιμοποιήσετε έναν διαφορετικό partitioner. Οι partitioners[5] χωρίζουν τον χώρο των κλειδιών έτσι ώστε ο κάθε reducer θα πάρει ένα κομμάτι του dataset. Αρκετά "δίκαιος" και λιγότερο πολύπλοκος partitioner είναι ο HashPartitioner, αλλά δεν μπορεί να εφαρμοστεί στην περίπτωσή μας, καθώς δεν διατηρεί την σειρά των αντικειμένων. Επομένως θα χρησιμοποιήσουμε range partitioning ο οποίος διατηρεί την σειρά των αντικειμένων. Range partitioning κάνει ο TotalOrderPartitioner του πακέτου org.apache.hadoop.mapreduce.lib.partition (οδηγίες εδώ: http://hadoop.apache.org/common/docs/current/api/org/apache/hadoop/mapreduce/lib/partition/TotalOrderPartitioner.html). Ο TotalOrderPartitioner μέσω της μεθόδου setPartitionFile ορίζει τα διαφορετικά κομμάτια στα οποία θα "σπάσει" το id space. Το partition file περιέχει εύρη τιμών στα οποία θα αντιστοιχηθεί και από ένας reducer. Το partition file θα πρέπει να δημιουργηθεί εκ των προτέρων με την μέθοδο της δειγματοληψίας. Με την μέθοδο αυτή τρέχουμε μια "μικρή" mapReduce δουλειά σε ένα δείγμα των δεδομένων και από αυτό κατασκευάζουμε το partition file (στην ουσία πρόκειται για μια προσέγγιση του ιστογράμματος). Πληροφορίες για να το κάνουμε αυτό

[5] http://en.wikipedia.org/wiki/Partition_(database)

βλέπουμε εδώ: http://chasebradford.wordpress.com/2010/12/12/reusable-total-order-sorting-in-hadoop/ Στην ουσία θα χρησιμοποιηθεί ο κώδικας της προηγούμενης ενότητας με μόνο έναν reducer ο οποίος θα δημιουργήσει το partition file.
Ζητείται η κατασκευή δυο partition files: στο πρώτο οι mapers θα διαβάζουν 50 samples ο καθένας, ενώ στο δεύτερο 1000. Εκτελέστε το πρόγραμμα που φτιάχνει τα ordered files χρησιμοποιώντας τον TotalOrderPartitioner με δυο διαφορετικά samplefiles. Έστω οι εκτελέσεις ordered_sample_50 και ordered_sample_1000. Τυπώστε το μέγεθος των αντικειμένων που υπάρχουν σε κάθε αρχείο για κάθε εκτέλεση.

2.5.3 Ερωτήσεις

- Πόσο χρόνο έκαναν να εκτελεστούν τα προγράμματα histogram_full, histogram_50 και histogram_1000?
- Πιο ιστόγραμμα προσεγγίζει καλύτερα το histogram_full?
- Με τι κόστος έγινε αυτή η προσέγγιση?
- Ποιο γράμμα έχει τα περισσότερα αποτελέσματα? Για ποιο λόγο συμβαίνει αυτό?
- Σε ποια εκτέλεση από τις ordered_sample_50 και ordered_sample_1000 τα αρχεία File1..File10 που προέκυψαν περιέχουν πιο ισοκατανεμημένο αριθμό κλειδιών?
- Ποια εκτέλεση από τις ordered_sample_50 και ordered_sample_1000 εκτελέστηκε πιο γρήγορα?

Hint5: Στο πρώτο μέρος χρησιμοποιείστε το wordcount χρησιμοποιώντας 27 reducers. Η ρύθμιση αυτή γίνεται με την εντολή jobConf.setNumReduceTasks(27) στην main συνάρτηση του hadoop job.

2.6 Υπολογισμός ποσοστού ερωτημάτων που μπορούν να απαντηθούν από την wikipedia

Μια πρόσφατη μελέτη[6] έδειξε κάτι που όλοι μας παρατηρούμε κατά την αναζήτηση άρθρων στο google: στο 99% των αναζητήσεων υπάρχει τουλάχιστον ένα άρθρο της Wikipedia στα πρώτα 10 αποτελέσματα. Σε αυτή την εργασία θα χρησιμοποιήσουμε το MapReduce framework για να κάνουμε μια παρόμοια ανάλυση.

Πιο συγκεκριμένα, ο τελικός στόχος είναι να δούμε ποιες λέξεις-κλειδιά των ερωτημάτων των χρηστών υπάρχουν σαν λέξεις και στους τίτλους των άρθρων της wikipedia. Στις περιπτώσεις που ισχύει αυτή η συνθήκη, μπορούμε να κάνουμε την απλοποιημένη θεώρηση ότι στα πρώτα results του χρήστη θα εμφανιστεί το αντίστοιχο άρθρο της Wikipedia σε μια μηχανή αναζήτησης. Ένα ερώτημα αποτελείται από μια ή περισσότερες λέξεις-κλειδιά, καθώς και ένας τίτλος ενός άρθρου της Wikipedia μπορεί να έχει περισσότερες από μια λέξεις.

2.6.1 Παράδειγμα

Έστω το ερώτημα q1:"Who is Einstein?". Επίσης, έστω το άρθρο της Wikipedia με τίτλο t1:"Albert_Einstein". Εφόσον το keyword "Einstein" υπάρχει και στο ερώτημα q1 και στο άρθρο με τίτλο t1 θεωρούμε ότι το άρθρο "Albert_Einstein" της Wikipedia θα είναι στα

[6] http://www.intelligentpositioning.com/blog/2012/02/wikipedia-page-one-of-google-uk-for-99-of-searches/

πρώτα results της αναζήτησης ”Who is Einstein?”. Σε αντίθετη περίπτωση, θεωρούμε ότι το ερώτημα δεν θα επιστρέψει αποτελέσματα από κανένα άρθρο της Wikipedia.

Για την εκτέλεση της εργασίας μπορείτε να χρησιμοποιήσετε όσα map/reduce βήματα θέλετε. Το τελικό ζητούμενο της ενότητας είναι ένας πίνακας της μορφής:

	Συνολικός αριθμός	**Ποσοστό %**
Ερωτήματα που περιέχουν στα αποτελέσματά τους άρθρα της wikipedia	Q_{exist}	$\frac{Q_{exist}}{Q_{exist}+Q_{not\_exist}} \cdot 100$
Ερωτήματα που δεν περιέχουν στα αποτελέσματά τους άρθρα της wikipedia	Q_{not_exist}	$\frac{Q_{not\_exist}}{Q_{exist}+Q_{not\_exist}} \cdot 100$

3 Εισαγωγή στις βάσεις NoSQL (HBase)

Σε αυτό το ερώτημα θα χρησιμοποιήσουμε τα σετ δεδομένων του ερωτήματος 2 με σκοπό να γεμίσουμε μια βάση NoSQL (συγκεκριμένα την HBase) με δεδομένα και να εκτελέσουμε ερωτήματα επάνω σε αυτά τα δεδομένα. Στην βάση θα εισάγουμε όλους τους τίτλους της wikipedia, και τα ερωτήματα που θα κάνουμε θα προέρχονται από το μικρό dataset της aol.

Στο πρώτο μέρος της άσκησης θα εισάγουμε τα δεδομένα στην βάση χρησιμοποιώντας το MapReduce framework. Στο δεύτερο μέρος της άσκησης θα χρησιμοποιήσουμε το API της hbase για να εκτελέσουμε απλά ερωτήματα στην βάση.

Ο τελικός στόχος είναι να μπορούμε να κάνουμε αναζητήσεις λέξεων κλειδιών στους τίτλους των άρθρων της wikipedia χρησιμοποιώντας την Hbase. Οι αναζητήσεις θα είναι της μορφής: ποια άρθρα της wikipedia έχουν στον τίτλο τους το keyword “X”? Για την απάντηση τέτοιου είδους ερωτημάτων θα δημιουργήσουμε ένα βοηθητικό ευρετήριο (index) που λέγεται ανεστραμμένο ευρετήριο (inverted index)[7].

3.1 Εισαγωγή των τίτλων της wikipedia στην Hbase.

Στην προηγούμενη άσκηση απλά μεταφέραμε το xml αρχείο με τους τίτλους της wikipedia στο hadoop. Παρόλο που η μεταφορά έγινε γρήγορα, δεν έχουμε την δυνατότητα να επεξεργαζόμαστε το αρχείο, κάτι που γενικά ισχύει για αρχεία κειμένου. Σε αυτή την περίπτωση, βολεύει η μεταφορά των δεδομένων σε μια βάση, έτσι ώστε να μπορούμε να έχουμε γρήγορη τυχαία πρόσβαση ή δυνατότητα ενημερώσεων των εγγραφών. Σε αυτό το ερώτημα θα μεταφέρουμε τους τίτλους των άρθρων της wikipedia στην Hbase.

Για τον σκοπό αυτό θα δημιουργήσουμε έναν πίνακα με το όνομα “content”. Σύμφωνα με το μοντέλο δεδομένων της hbase, κάθε άρθρο μπορεί να αποθηκευτεί σε μια διαφορετική γραμμή (row), σε ένα ξεχωριστό κελί στον πίνακα content. Έτσι, για κάθε άρθρο θα δημιουργηθεί ένα κελί, το οποίο θα έχει ένα αναγνωριστικό γραμμής (row key) και μια τιμή (value). Η τιμή του κελιού θα είναι ο τίτλος του άρθρου. Το αναγνωριστικό του κελιού θα είναι ένα μοναδικό (unique) id με το οποίο θα μπορούμε να το ανακτήσουμε. Στην άσκηση

[7] http://en.wikipedia.org/wiki/Inverted_index

θα χρησιμοποιήσουμε σαν αναγνωριστικό το md5hash του τίτλου. Έτσι, πχ ο τίτλος Bill_gates θα αποθηκευτεί στον πίνακα content σαν ένα κελί με την τιμή "Bill_Gates" και αναγνωριστικό md5("Bill_Gates")="4243058b3ef87cdc2a251a3f14717ec0". Με αυτόν τον τρόπο, μπορούμε να ανακτήσουμε την τιμή του κελιού εάν γνωρίζουμε το αναγνωριστικό της γραμμής με την get(key) εντολή της hbase. Το αναγνωριστικό δεν έχει κάποια ιδιαίτερη σημασία, απλά χρειάζεται να είναι μοναδικό για κάθε άρθρο.

Για την δημιουργία του πίνακα θα χρησιμοποιήσετε τον μηχανισμό bulk loading[8] της hbase, ο οποίος είναι πιο γρήγορος από την απλή εκτέλεση διαδοχικών Put commands. Ο μηχανισμός bulk loading πρόκειται στην ουσία για μια εργασία mapReduce η οποία παίρνει σαν input ένα set αρχείων από το hdfs και σαν output δημιουργεί regions της HBase, οι οποίες κατόπιν εισάγονται στην βάση. Τα regions της hbase στην ουσία αποτελούνται από αρχεία στο hdfs. Ο αριθμός των αρχείων που θα αποτελέσουν τα regions ορίζονται από τον αριθμό των reducers που θα βάλετε. Επίσης, σαν output του job θα πρέπει να ορίσετε την κλάση HFileOutputFormat. Στο παρακάτω link περιγράφεται η διαδικασία εισαγωγής των δεδομένων από το hdfs σε έναν πίνακα στην HBase. Ο πίνακας και τα column families θα πρέπει να προϋπάρχουν στην HBase. Επομένως θα δημιουργήσετε έναν κενό πίνακα μαζί με τα column families που θέλετε με το hbase shell, όπως είδαμε στο φροντιστήριο. Χρήσιμο link είναι και αυτό [9] το οποίο τρέχει από command line την εντολή LoadIncrementalHFiles που έχουν δημιουργηθεί. Για να τρέξετε την εντολή μέσα από κώδικα, μπορείτε να τρέξετε αυτό:

```
ToolRunner.run(new LoadIncrementalHFiles(HBaseConfiguration.create())
```

Ερώτηση: Γιατί ο μηχανισμός bulk import είναι πιο γρήγορος από διαδοχικά Put για κάθε αντικείμενο? Πως εξηγείται αυτό λαμβάνοντας υπόψη την αρχιτεκτονική της HBase κατά την εισαγωγή νέων αντικειμένων?

3.2 Εξαγωγή ανεστραμένου ευρετηρίου inverted index των λέξεων κλειδιών.

Αφού έχουμε δημιουργήσει τον πίνακα content, τώρα μπορούμε να δημιουργήσουμε το ανεστραμμένο ευρετήριο (έστω πίνακας index). Σύμφωνα με τον ορισμό του ανεστραμμένου ευρετηρίου, η δομή αυτή περιέχει την αντιστοίχηση περιεχομένου (πχ λέξεις κλειδιά) στην τοποθεσία που αυτό έχει εντοπιστεί (πχ συγκεκριμένα έγγραφα). Τα ανεστραμμένα ευρετήρια χρησιμοποιούνται στην αναζήτηση κειμένου (full text search) και ίσως το μεγαλύτερο ανεστραμμένο ευρετήριο είναι αυτό που διατηρεί το google για την εκτέλεση των αναζητήσεων.
Για να καταλάβουμε καλύτερα τα ανεστραμμένα ευρετήρια, ας θεωρήσουμε το αντίστοιχο παράδειγμα. Έστω ότι έχουμε τα παρακάτω κείμενα:

T_0="it is what it is"
T_1="what is it"
T_2="it is a banana"

[8] http://hbase.apache.org/book/arch.bulk.load.html

[9] http://hbase.apache.org/book/ops_mgt.html#completebulkload

Το ανεστραμμένο ευρετήριο με τις εμφανίσεις των λέξεων σε κείμενα μπορεί να είναι της μορφής:

"a": {2}
"banana": {2}
"is": {0, 1, 2}
"it": {0, 1, 2}
"what": {0, 1}

Χρησιμοποιώντας το ευρετήριο μπορούμε να βρούμε γρήγορα σε ποια κείμενα βρίσκεται μια συγκεκριμένη λέξη: πχ η λέξη banana υπάρχει μόνο στο κείμενο 2, ενώ η λέξη it υπάρχει στα κείμενα 0,.1, 2.

Για την δημιουργία του ανεστραμμένου ευρετηρίου θα χρησιμοποιήσουμε πάλι την δυνατότητα bulk loading. Η mapreduce εργασία που θα δημιουργήσει τον πίνακα index θα πάρει σαν input τον πίνακα content. Στην ουσία, η mapreduce εργασία θα είναι το παράδειγμα wordcount που χρησιμοποιήσαμε και στα προηγούμενα ερωτήματα, **με την διαφορά ότι οι mapers θα κάνουν emmit μαζί με την λέξη και το id του κειμένου στο οποίο βρέθηκε** (αντί για "1" που κάνει emit το wordcount). Κατόπιν, οι reducers θα συλλέγουν για κάθε μοναδική λέξη τα ids των κειμένων που την περιέχουν, και θα δημιουργούν την γραμμή με τις αντιστοιχήσεις. Μετά την εκτέλεση αυτού του ερωτήματος θα υπάρχουν 2 πίνακες στην hbase: ο πίνακας content που θα περιέχει τους τίτλους των άρθρων στην wikipedia και ο πίνακας index που θα περιέχει το ανεστραμμένο ευρετήριο για την εκτέλεση ερωτημάτων.

3.3 Εκτέλεση ερωτημάτων

Σε αυτό το ερώτημα θα χρησιμοποιήσουμε τον client της hbase για να εκτελέσουμε ερωτήματα στους τίτλους της wikipedia. Οδηγίες χρήσης του client μπορούμε να βρούμε εδώ[10]

Σε γενικές γραμμές, δημιουργούμε ένα αντικείμενο τύπου HTable, και τα ερωτήματα γίνονται με την χρήση της κλάσης Get:

```
Get g = new Get(Bytes.toBytes("keyword"));
Result r = table.get(g);
```

και ψάχνουμε το αποτέλεσμα result ως εξής:

```
byte [] value = r.getValue(Bytes.toBytes("myLittleFamily"),
    Bytes.toBytes("someQualifier"));
String valueStr = Bytes.toString(value);
System.out.println("GET: " + valueStr);
```

Για την δημιουργία των ερωτημάτων θα χρησιμοποιήσουμε το αρχείο με τα δημοφιλή keywords των προηγούμενων ερωτημάτων. Για να το κάνουμε αυτό, θα αποθηκεύσουμε την λίστα με τα 1000 ποιο δημοφιλή keywords, και θα διαλέγουμε με τυχαίο τρόπο ένα κάθε φορά. Κάντε συνολικά 10000 ερωτήματα στον πίνακα index. Πόσα ερωτήματα επιστρέφουν αποτελέσματα?

[10] http://hbase.apache.org/apidocs/org/apache/hadoop/hbase/client/package-summary.html

4 Κλιμακωσιμότητα Hadoop MapReduce και HBase

Σε αυτή την ενότητα θα μελετήσουμε πως επηρεάζεται ο χρόνος εκτέλεσης της επεξεργασίας μεγάλου όγκου δεδομένων σε διαφορετικά μεγέθη συστοιχιών Hadoop και HBase. Για τον σκοπό αυτό θα χρησιμοποιήσουμε μια προ-εγκατεστημένη συστοιχία Hadoop και HBase στον okeano που θα αποτελείται από έναν μεγάλο αριθμό υπολογιστικών κόμβων.

Τα δεδομένα που θα χρησιμοποιήσουμε είναι του ίδιου τύπου με τα δεδομένα της ενότητας 1.1 και θα είναι ανεβασμένα στο HDFS εκ των προτέρων.

4.1 Υπολογισμός ποσοστού ερωτημάτων που μπορούν να απαντηθούν από την wikipedia

Σε αυτή την ενότητα θα εκτελέσουμε τον κώδικα του ερωτήματος 2.6. Θα χρησιμοποιήσουμε 3 διαφορετικά μεγέθη συστοιχιών για να εκτελέσουμε τον κώδικα MapReduce στο ίδιο dataset. Στην αναφορά φτιάξτε έναν πίνακα και ένα διάγραμμα που να δείχνει πως επηρεάζεται ο χρόνος εκτέλεσης από το μέγεθος του cluster. Τι παρατηρείτε?

4.2 Εξαγωγή ανεστραμένου ευρετηρίου inverted index των λέξεων κλειδιών των άρθρων της wikipedia

Σε αυτή την ενότητα θα εκτελέσουμε τον κώδικα του ερωτήματος 3.2. Ο πίνακας στην HBase με τα άρθρα της Wikipedia θα είναι ήδη έτοιμος. Καλείστε να εκτελέσετε τον κώδικα που εξάγει το ανεστραμμένο ευρετήριο με 3 διαφορετικά μεγέθη συστοιχιών. Στην αναφορά φτιάξτε έναν πίνακα και ένα διάγραμμα που να δείχνει πως επηρεάζεται ο χρόνος εκτέλεσης από το μέγεθος του cluster. Τι παρατηρείτε?

5 Διαδικαστικά

- Η άσκηση θα υλοποιηθεί είτε ατομικά, είτε σε ομάδες 2 ατόμων.
- Η παράδοση θα γίνει στο mail atds@cslab.ntua.gr. Το θέμα του mail θα είναι "Εξαμηνιαία Εργασία Προχωρημένα Θέματα Βάσεων Δεδομένων".
- Η προθεσμία παράδοσης είναι την Δευτέρα 14 Απριλίου 2014.
- Θα ακολουθήσουν οδηγίες για την εκτέλεση των ερωτημάτων της ενότητας 4.
- Η παράδοση θα αποτελείται από:
    - Μια σύντομη αναφορά όπου θα περιγράφετε την μεθοδολογία που ακολουθήσατε (όχι κώδικας εδώ).
    - Ψευδοκώδικας για τα προγράμματα Map/Reduce που χρησιμοποιήσατε για κάθε κομμάτι της άσκησης. Ο ψευδοκώδικας θα δείχνει εποπτικά τα key/values που παίρνει η συνάρτηση map, την επεξεργασία που τους κάνει, τα key/values που κάνει emit στην συνάρτηση reduce, και την επεξεργασία που κάνει η reduce (σαν τον ψευδοκώδικα του wordcount).
    - Links στο hdfs site όπου έχετε βάλει τα datasets καθώς και τα ενδιάμεσα/τελικά αποτελέσματα.
    - Ένα zip file με τον κώδικα.
    - Ένα zip file με τα log-files των εργασιών MapReduce από τις οποίες βγήκαν τα αποτελέσματα.