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 2017-2018, 9ο Εξάμηνο ΗΜ&ΜΥ

Β. Καντερέ, Κ. Τζαμαλούκας, Ι. Κωνσταντίνου

Εξαμηνιαία Εργασία

Εισαγωγή στο MapReduce

1 Εισαγωγή

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

Στην 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.

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 χρήστης. Σε περίπτωση που ο χρήστης δεν επέλεξε κάποιο result, 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:

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

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.blocksize</name>
      <value>33554432</value>
</property>
```

Εάν αυτό δεν λειτουργεί σωστά, δοκιμάστε την παρακάτω εντολή στην main συνάρτηση του κάθε MapReduce Job:

conf.setLong("mapreduce.input.fileinputformat.split.maxsize", 33554432);

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 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 hdfs dfs

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

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

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, εβδομάδα και μήνα. (10%)

Σε αυτή την φάση, ζητούνται τρία MapReduce προγράμματα που θα υπολογίζουν τις τιμές για την εξαγωγή 3 γραφημάτων που θα δείχνουν για κάθε διαφορετική μέρα, εβδομάδα και μήνα αντίστοιχα της συνολικής χρονικής περιόδου, τον αριθμό των αναζητήσεων που έγιναν για την μέρα την εβδομάδα ή τον μήνα αυτόν. 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, εβδομάδα ή μήνα.

2.2 Υπολογισμός ποσοστού «επιτυχών» και «ανεπιτυχών» αναζητήσεων. (10%)

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. (10%)

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. (20%)

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 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.

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. (30%)

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 "ίσα" κομμάτια, όπου ο πρώτος reducer θα έπαιρνε πχ τις λέξεις που ξεκινάνε από [A..C], ο 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

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

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 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

[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

```
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 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 mappers θα διαβάζουν 10 samples ο καθένας, ενώ στο δεύτερο 1000. Εκτελέστε το πρόγραμμα που φτιάχνει τα ordered files χρησιμοποιώντας τον TotalOrderPartitioner με δυο διαφορετικά sample files. Έστω οι εκτελέσεις ordered_sample_10 και ordered_sample_1000. Τυπώστε το μέγεθος των αντικειμένων που υπάρχουν σε κάθε αρχείο για κάθε εκτέλεση.

---

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

2.5.3 Ερωτήσεις

- Πόσο χρόνο έκαναν να εκτελεστούν τα προγράμματα histogram_full, histogram_10 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_10 και ordered_sample_1000 τα αρχεία File1..File10 που προέκυψαν περιέχουν πιο ισοκατανεμημένο αριθμό κλειδιών?
- Ποια εκτέλεση από τις ordered_sample_10 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 (20%)

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 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 %**

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

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 Διαδικαστικά

- Η άσκηση θα υλοποιηθεί είτε ατομικά, είτε σε ομάδες 2 ατόμων.
- Η παράδοση θα γίνει στο mycourses site.
- Η προθεσμία παράδοσης είναι την Παρασκευή 22 Δεκεμβρίου 2017 23:59.
- 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.