Ατομική Διπλωματική Εργασία

Characterization of Big Data Analytics Application

Χρυστάλλα Τσουτσούκη

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Τίτλος Ατομικής Διπλωματικής Εργασίας

Χρυστάλλα Τσουτσούκη

Επιβλέπων Καθηγητής
Γιαννάκης Σαζείδης

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Γιαννάκη Σαζεΐδη. ο οποίος ήταν πάντοτε πρόθυμος να με βοηθήσει και να ακούσει την πρόοδο μου σε όλη την διάρκεια της διπλωματικής μου εργασίας. Επίσης κατάφερα να αποκομίσω από αυτόν ένα ορθολογικό τρόπο με τον οποίο πρέπει να δουλεύω, κάτι που θα χρησιμοποιήσω και στην μετέπειτα ζωή μου.

Επίσης τον Ζαχαρία Χατζιλάμπρου για την βοήθεια που μου έδωσε όποτε την χρειάστηκα.

Περίληψη

Η αύξηση των δεδομένων που έχουμε σήμερα στην κατοχή μας , μας δημιούργησε την ανάγκη να τα επεξεργαστούμε ώστε να μάθουμε μέσα από αυτά. Η σημαντικότητα της ανάλυσης των δεδομένων επηρεάζει τόσο τις εταιρείες που τους παρέχει την επιπλέον χρήσιμη γνώση αλλά ταυτόχρονα κίνησε το ενδιαφέρων και των ερευνητών που ασχολούνται με τα τεχνικά κομμάτια. Οι εγκαταστάσεις στις οποίες επεξεργάζονται αυτά τα δεδομένα ονομάζονται Data centers και χρειάζονται αρκετό κόστος για να δημιουργηθούν και να επιτηρούνται. Γι' αυτούς τους λόγους η κατανόηση των εφαρμογών που τρέχουν έτσι ώστε να βρίσκουν λύσεις μείωσης του κόστους είναι αρκετά σημαντική.

Με σκοπό να προτείνουμε μια καλύτερη λύση σε θέματα ανάλυσης δεδομένων αναπτύξαμε αυτή την διπλωματική η οποία κυμαινόταν σε τρία επίπεδα. Το πρώτο ήταν η πολύ καλή κατανόηση του λογισμικού μέρους. Συγκεκριμένα η αντίληψη εις βάθος του πως δουλεύει ο αλγόριθμος(Naïve Bayes) , τα frameworks (Hadoop), τις βιβλιοθήκες (mahout) που χρησιμοποιούμε ως οντότητες από μόνες τους, αλλά επίσης και πως το ένα προσαρμόζεται πάνω στο άλλο.

Ακολούθως είχαμε να βρούμε τα κατάλληλα δεδομένα που θα μας εξυπηρετούσαν στην δουλειά που θέλουμε να κάνουμε αφού όπως θα αναφερθεί και στη συνέχεια η εύρεση αντιπροσωπευτικού δείγματος δεδομένων δεν είναι και τόσο απλό. Αφού βρήκαμε τα δεδομένα τη συνέχεια έπρεπε να τα καταλάβουμε και να τα προσαρμόσουμε, ώστε να μπορούν να χρησιμοποιηθούν από την εφαρμογή σαν είσοδο. Αξίζει να σημειωθεί ότι οποιαδήποτε επεξεργασία των δεδομένων πριν τα χρησιμοποιήσουμε στην εφαρμογή θεωρείται ένα από το κομμάτι της ανάλυσης.

Τέλος χρησιμοποιώντας όλες τις γνώσει που αποκομίσαμε από τα προηγούμενα στάδια κάναμε διάφορα πειράματα όπως διαφορετικά data sizes , χρησιμοποίηση λιγότερων ή περισσότερων πυρήνων. Η σημαντική παρατήρηση που έγινε μετά τα πειράματα είναι ότι αν και αρχικά περιμέναμε τα Big Data Analytics να είναι memory –intensive μας απέδειξε η συγκεκριμένη εφαρμογή ότι είναι περισσότερο cpu intensive. Επίσης μας έδειξε πως σε τέτοιου είδους εφαρμογές τα δεδομένα περνούν από πολλά στάδια και έχουν διαφορετικές συμπεριφορές ακόμη και κατά την εκτέλεση μιας λειτουργίας κάνοντας δυσκολότερο αλλά και πιο ενδιαφέρων τη δουλειά των ερευνητών που θα θέλουν να προσφέρουν την καλύτερη δυνατή λύση για αυτού του είδους τις εφαρμογές.

Περιεχόμενα.

Κεφάλαιο 1 ***Εισαγωγή***

1.1 Κίνητρο

Βρισκόμαστε στον 21^{ο} αιώνα όπου σχεδόν το 40% του πληθυσμού της γης χρησιμοποιεί το διαδίκτυο, με αποτέλεσμα ο αριθμός των δεδομένων να αυξάνεται ραγδαία μέρα με την μέρα. Τα δεδομένα λοιπόν αποτελούν μια σημαντική πηγή γνώσης, μπορούν να δώσουν δύναμη σε μία εταιρεία που θέλει να ανταγωνιστεί και να προσφέρει το κάτι παραπάνω, να απαντήσει ερωτήματα που αφορούσαν το παρελθόν αλλά να προβλέψει το μέλλον.

Το μεγάλο μέγεθος πληροφοριών , ο γρήγορος ρυθμός με τον οποίο παράγονται , τα πολλά διαφορετικά είδη που μπορούμε να έχουμε καθώς και η ανάγκη μείωσης του χρόνου για συλλογή των πραγματικά χρήσιμων δεδομένων, αποτελούν τα χαρακτηριστικά της ομάδας δεδομένων που ονομάζεται Big Data. Για να καταφέρουμε να εκμεταλλευτούμε όλα τα καλά που μας προσφέρουν τα BigData παρουσιάστηκε η ανάγκη ανάπτυξης νεών υποδομών, framework και αλγορίθμων. Για να μπορέσουμε να αξιοποιήσουμε όλα αυτά τα δεδομένα χρησιμοποιούμε κατάλληλες μεθόδους επεξεργασίας και ανάλυσης που θα μας δώσουν την κρυμμένη γνώση γρήγορα.

Η ευρεία χρήση των αλγόριθμων ανάλυσης δεδομένων σε πολλές εταιρείες κατέστησε την ανάγκη κατανόησης της συμπεριφοράς τους όσον αφορά τους Data centers που στεγάζονται μέσα σε αυτές τις εταιρείες και αποτελούν ένα από τα σημαντικότερα έξοδα της.

1.2 Συνεισφορά

Τα αποτελέσματα αυτής της εργασίας μπορούν να δώσουν μια βαθύτερη γνώση και αντίληψη στο πως δουλεύουν κάποιοι από τους αλγόριθμους ανάλυσης δεδομένων και πως αυτά επηρεάζουν το σύστημα , την απόδοση και το υλικό ενός υπολογιστή . Μια μικρή αλλαγή σε ένα κομμάτι του συστήματος μπορεί να είναι σημαντικό τόσο για την επίδοση όσο και για το κόστος.

1.3 Οργάνωση Εργασίας

Στο κεφάλαιο 2 θα αναφερθούμε σε γενικές ορολογίες που είναι χρήσιμες για την κατανόηση καλύτερα εννοιών που χρησιμοποιούνται αργότερα. Στη συνέχεια , στο κεφάλαιο τρία κάνουμε αναφορά σε σχετικές εργασίες που φάνηκαν χρήσιμες για την διεκπεραίωση της δικής μας. Ακολούθως θα επικεντρωθούμε σε θέματα ανάλυσης δεδομένων όπως για παράδειγμα χαρακτηριστικά αυτών καθώς και εξήγηση του αλγόριθμου που χρησιμοποιήσαμε εμείς. Αφού θα έχουμε καλύψει αρκετά τα θέματα λογισμικού θα εξηγήσουμε την αρχιτεκτονική πάνω στην οποία τρέξαμε που δεν είναι άλλη το Hadoop της Apache σε συνδυασμό με την βιβλιοθήκη Mahout. Τέλος στο κεφάλαιο 6 θα αναφερθούμε στην πειραματική μας δουλειά και τα συμπεράσματα στα οποία καταλήξαμε.

Κεφάλαιο 2 Τεχνικό Υπόβαθρο

2.1 Big Data

Γενικά με τον όρο Big Data αναφερόμαστε στα δεδομένα τα οποία δεν μπορούν να επεξεργαστούν ή να αναλυθούν εφαρμόζοντας παραδοσιακές μεθόδους ή εργαλεία [1]. Για να μπορέσουμε σήμερα να έχουμε μια πιο ξεκάθαρη ιδέα για το τι είναι Big Data ορίστηκαν κάποια χαρακτηριστικά τα οποία είναι γνωστά ως τα "4Vs" των Big Data· Volume , Variety, Velocity και Veracity.

Λόγω της ευρείας χρήσης του διαδικτύου, ο όγκος των δεδομένων που είναι αποθηκευμένα σήμερα στους Data Centers είναι τεράστιος, για παράδειγμα η Facebook είχε αναφέρει πως έχει πάνω από 100PB δεδομένα media αποθηκευμένα.[2] . Αν συνεχίσουμε με τους ίδιους ρυθμούς αναμένεται πως θα φτάσουμε τα zettabytes (ZB) το 2020. Με το χαρακτηριστικά Volume λοιπόν αναφερόμαστε στο μεγάλο μέγεθος των δεδομένων. Το τι θεωρείται μεγάλος αριθμός δεδομένων ,είναι κάτι που διαφοροποιείται μέρα με την μέρα.

Το επόμενο χαρακτηριστικό είναι το Variety. Με την έκρηξη των έξυπνων συσκευών και των πολλών εφαρμογών οι πληροφορίες που συλλέγουμε έχουν γίνει περίπλοκες λόγω του ότι δεν περιέχουν μόνο τα παραδοσιακά δεδομένα αλλά επιπρόσθετα ακατέργαστα, δομημένα ή μη δομημένα από κάθε είδους πηγή. Επομένως, αν θέλουμε να αναλύσουμε αυτά τα δεδομένα πρέπει να είμαστε σε θέση να επεξεργαστούμε όλους τους διαθέσιμους τύπους. Πράγμα που οι παραδοσιακές μέθοδοι δεν είναι σε θέση να το κάνουν.

Εκτός από τον όρο μεγάλο (volume) και διαφορετικό(Variety), σημαντικό ρόλο παίζει ο ρυθμός με τον οποίο τα δεδομένα παράγονται και η ταχύτητα που πρέπει να τα διαχειριζόμαστε για να δίνουμε τα αποτελέσματα που θέλουμε την χρονική στιγμή που τα θέλουμε.

Και τέλος έχουμε τον όρο Veracity (φιλαλήθεια) [3] ή διαφορετικά Accuracy (εγκυρότητα). Σε μια πρόσφατη αναφορά της Ventana αναφέρθηκε ότι σε κάθε διαδικασία ανάλυσης δεδομένων , το 40-60% του χρόνου καταναλώνεται στην προετοιμασία των δεδομένων. Για παράδειγμα διαγραφή διπλότυπων, μείωση κενών , ένωση δεδομένων και πολλά άλλα. [4][15]

2.2 Data Science and Data Analytics

Επιστήμη και ανάλυση δεδομένων είναι η συλλογή , οργάνωση και ανάλυση δεδομένων με σκοπό την εξαγωγή γνώσης μέσα από αυτά. Μπορούμε να πούμε πως η ανάλυση των δεδομένων αποτελεί ένα από τα σημαντικότερα θέματα μιας επιχείρησης αφού μια καλή μελέτη μπορεί να τους προσφέρει χρήσιμες πληροφορίες όπως η προτιμήσεις των πελατών , προβλέψεις για το μέλλων κ.α. οδηγώντας σε ανταγωνιστικό πλεονέκτημα την εταιρεία αυτή.

Τα Data Analytics χωρίζονται σε Descriptive data analytics ,αναφέρονται στην ανάλυση δεδομένων του παρελθόντος με σκοπό να βρουν κρυμμένα μοτίβα και χρήσιμες πληροφορίες. Αποτελούν το 80% της ανάλυσης που γίνεται σε δεδομένα σήμερα. Η επόμενη κατηγορία είναι τα Predictive data analytics , στη συγκεκριμένη κατηγορία χρησιμοποιούμε δεδομένα του παρελθόντος και με στατιστικούς αλγόριθμους μπορούμε να υπολογίσουμε το τι θα συμβεί στο μέλλον. Τέλος μια κατηγορία η οποία πρόσφατα μπέικε στο προσκήνιο είναι η Prescriptive που είναι ένα είδος Predictive αλλά επιπρόσθετα δίνει και ένα outocome για κάθε μια από τις προβλέψεις του παίρνοντας με κάποιο τρόπο και αποφάσεις.

Μια άλλη κατηγοριοποίηση που των Data analytics είναι τα iterative και interactive jobs. Στο πρώτο κατά την εκτέλεση του αλγόριθμου φορτώνονται δεδομένα στη μνήμη τα επεξεργαζόμαστε και στη συνέχεια ξαναφορτώνουμε τα ίδια δεδομένα για την επόμενη επεξεργασία . Η διαφορά με το επόμενο βρίσκεται στην πρόσβαση στο δίσκο αφού το $2^{ο}$ αναφέρεται στις εργασίες που φορτώνουν τα δεδομένα στη μνήμη και επαναλαμβάνουν επεξεργασία σε αυτά και τα παράγωγα τους.

2.3 Data Centers

Μία από τις κυρίαρχες έννοιες που επικρατούν στις μέρες μας είναι το cloud computing το οποίο αναφέρεται στην online παροχή υπηρεσιών και πόρων στους χρήστες ανά το παγκόσμιο. Οι εγκαταστάσεις που στεγάζουν τους υπολογιστικούς πόρους πάνω στα οποία διενεργούνται οι λειτουργίες του νέφους ονομάζονται data centers.

Η μεγάλη χρήση , οι πολλές εφαρμογές και ο τεράστιος και περίπλοκος όγκος δεδομένων που έχουν να επεξεργαστούν αυτά τα υπολογιστικά συστήματα απαιτούν τεράστιες ποσότητες υπολογιστικών πόρων , χώρου, ενέργειας καθώς και νέων κατάλληλων επεξεργαστών.

Αδήριτη ανάγκη είναι να μπορούμε με βάση τις απαιτήσεις των εφαρμογών μας να διαμορφώνουμε κατάλληλα το υλικό μας (δομή, ποιότητα κ.α.) ούτως ώστε να μειώνεται όσο το δυνατόν το κόστος και να αυξάνεται η απόδοση.

2.4 Total Cost of Ownership (TCO)

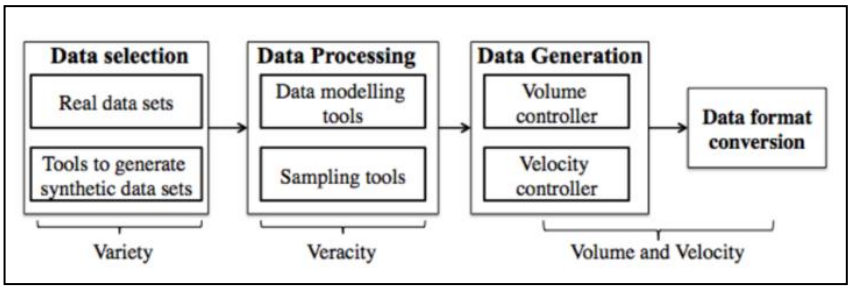
Το να δημιουργήσουμε και να συντηρήσουμε ένα data center απαιτεί εκατομμύρια δολάρια και αποτελεί ένα από τα σημαντικότερα έξοδα μιας εταιρείας .Όταν μιλάμε για total cost of ownership(TCO) αναφερόμαστε σε μια οικονομική ορολογία που εκτιμά το άμεσο και έμμεσο κόστος ενός αντικειμένου ή συστήματος δίνοντας μια καλύτερη αντίληψη του χρήστη για το πραγματικό κόστος. Βασικοί παράγοντες μπορεί να είναι το κόστος ανέγερσης του κτηρίου, η ενέργεια που θα καταναλώνεται , το κόστος του μηχανισμού cooling κ.α.

Κεφάλαιο 3 Σχετική Εργασία

Λόγω του ότι ο τομέας με τον οποίο ασχολείται η διπλωματική μου εξαρτάται από πολλές πιο μικρές υποκατηγορίες διάβασα άρθρα που θίγουν διάφορους τομείς. Επομένως μπορούμε να τα κατατάξουμε σε 4 διαφορετικές κατηγορίες. Οι κατηγορίες αυτές είναι: Big Data, Benchmark Suites, Data analytics και Frameworks. Θα αναφέρω στοιχεία μόνο για τις πρώτες δύο πρώτες αφού για τις δύο τελευταίες γίνονται αναφορές στο κείμενο.

3.1 Μεγάλα Δεδομένα.

Στον ερευνητικό τομέα ένα από τα βασικά προβλήματα που αντιμετωπίζουμε είναι η εύρεση δεδομένων εισόδου σε μια εφαρμογή τα οποία να είναι όσο πιο ρεαλιστικά γίνεται έτσι ώστε να μας δίνουν την πραγματική εικόνα για το πώς δουλεύει μια εφαρμογή. Δυστυχώς τα ρεαλιστικά δεδομένα δύσκολα δίνονται στην ερευνητική κοινότητα αφού περιέχουν προσωπικά στοιχεία. Λύση στο πρόβλημα αυτό έρχονται να δώσουν μοντέλα τα οποία παράγουν μη ρεαλιστικά δεδομένα αλλά με τα χαρακτηριστικά αυτών όπως φαίνεται στην εικόνα 3.1



Εικόνα 3.1 Μοντέλο παραγωγής δεδομένων

Με τα μεγάλα δεδομένα όμως αυτό είναι πρόβλημα, κανένα μέχρι τώρα μοντέλο παραγωγής δεδομένων δεν μπορεί να δώσει τα σωστά χαρακτηριστικά που έχουν τα πραγματικά μεγάλα δεδομένα. Στον πίνακα μπορούμε να δούμε κάποια benchmark suite και τι χαρακτηριστικά των big data καλύπτουν με τα δεδομένα που παρέχουν. [10][11]

Benchmark efforts	Volume	Velocity	Variety (data sources)	Veracity
Hibench [12]	Partially scalable	Un-controllable	Texts	Un-considered
GridMix [4]	Scalable	Un-controllable	Texts	Un-considered
PigMix [6]	Scalable	Un-controllable	Texts	Un-considered
YCSB [9]	Scalable	Un-controllable	Tables	Un-considered
Performance benchmark [15]	Scalable	Un-controllable	Tables, texts	Un-considered
TPC-DS [11]	Scalable	Semi-controllable	Tables	Partially Considered
BigBench [11]	Scalable	Semi-controllable	Texts, web logs tables	Partially Considered
LinkBench [17]	Partially scalable	Semi-controllable	Graphs	Partially Considered
CloudSuite [10]	Partially scalable	Semi-controllable	Texts, graphs, videos, tables	Partially Considered
BigDataBench [19]	Scalable	Semi-controllable	Texts, resumes, graphs, tables	Considered

Πίνακας 3.1

3.2 Benchmark Suites

Για να μπορέσουμε να αναλύουμε τις συμπεριφορές κάποιον προγραμμάτων κάνουμε χρήση των λεγόμενων benchmarks τα οποία είναι ένα σύνολο από προγράμματα ή άλλες λειτουργίες τα οποία έχουν σκοπό να αναδείξουν τον τρόπο που λειτουργεί ένα αντικείμενο, την απόδοση του κ.α. Για να το πετύχουν αυτό συμπεριλαμβάνουν προγράμματα το κάθε ένα από τα οποία έχει μια συγκεκριμένη ιδιότητα στο υλικό του υπολογιστή. Για παράδειγμα κάποια μπορεί να έχουν πολλές προσβάσεις στη μνήμη άλλα να έχουν μεγάλη επεξεργαστική χρήση.

Για τα μεγάλα δεδομένα υπάρχουν αρκετά benchmarks τα οποία εμφανίστηκαν τα τελευταία χρόνια. Κάποια από αυτά είναι το BigDataBench[12] , MineBench[13],HiBench[14]. Μέσα από τα άρθρα που δημοσίευσαν μπόρεσα να βρω σημαντικές πληροφορίες για χαρακτηριστικά αλγορίθμων και ιδιαίτερα το άρθρο για το MineBench ήταν ιδιαίτερα ενδιαφέρον. Ακόμη και να μην χρησιμοποιήσει κανείς τις εφαρμογές αυτών των Benchmark Suite είναι πολύ ενδιαφέρον το γεγονός και μόνο να κοιτάξει τις μεθοδολογίες που χρησιμοποιούν . Επίσης τα αποτελέσματα που εξάγουν και οι παρατηρήσεις που κάνουν μπορούν να γίνουν σημείο αναφοράς, επαλήθευσης και διάψευσης για κάθε ερευνητή.

Πιο πολύ ασχολήθηκα με το Cloud suite Benchmark Suite [16] το οποίο σε ένα από τα άρθρα που το χρησιμοποιούν , συγκρίνει τους επεξεργαστές τους οποίους χρησιμοποιούμε και τα Scale-Out Workloads. Σε αντίθεση με την κατανεμημένη, χωρίς εξαρτήσεις συμπεριφορά των workloads , οι επεξεργαστές προσπαθούν να βελτιώσουν την απόδοση ενός υπολογιστή, χρησιμοποιώντας μεγάλες συχνότητες, prefetching τεχνικές και άλλα πολλά.

Κάποιες τεχνικές ανάλυσης επίδοσης και της μηχανής που σύλλεξα από τέτοια είδους άρθρα είναι οι ακόλουθες:

- Use operf –a profiling tool[16]
- Use HiTune- distributed profiling tool for Hadoop framework to monitor the progress of high level data flow graph of each workload [17]
- Collecting data of timeline-based utilization of system resources e.g. disk , CPU , memory and network
- Use perf Linux profile tool that measure performance counters

Και κάποιες μέθοδοι:

Μετρικές μικροαρχιτεκτονικού επιπέδου:

- Instructions per cycle(IPC)
- L1/L2/Last level instruction cache miss ratio
- Branch miss prediction per instruction
- Off-chip bandwidth

Μέθοδοι ανάλυσης:

- Clustering techniques (Ανάλυση ομοιότητας)
- Μείωση παραμέτρων χωρίς να χάνονται τα χαρακτηριστικά των δεδομένων.

Κεφάλαιο 4 Data Analytics Benchmark

4.1 Machine Learning

Το machine learning είναι μια κατηγορία αλγορίθμων που είναι data –driven , σε αντίθεση με άλλα τους άλλους αλγόριθμους είναι τα δεδομένα που λένε αν μια απάντηση είναι καλή ή όχι. Για παράδειγμα ένας αλγόριθμος Handwriting Recognition,που προσπαθεί να καταλάβει τι γράφει ένας άνθρωπος χρησιμοποιώντας τον παραδοσιακό τρόπο της πένας και όχι τον ηλεκτρονικό υπολογιστή. Αυτός ο αλγόριθμος δεν έχει κάποιο κώδικα που να του λέει πως μοιάζει το α, β, γ…αλλά μαθαίνει από τα παραδείγματα δηλαδή μαθαίνει από τα δεδομένα που του εισάγουμε. Άρα καλά δεδομένα και καλή μέθοδος μάθησης θα μας δώσουν καλά αποτελέσματα διαφορετικά όχι. Αυτό το είδος μάθησης είναι το Supervised learning που σημαίνει ότι πρέπει να βάλουμε «ετικέτες» στα παραδείγματα μας. Όταν θα έχουμε τα αποτελέσματα μπορούμε να ξέρουμε αν αυτά που βρήκαμε είναι σωστά , αφού έχουμε τις ετικέτες.

Στο unsupervised learning προσπαθούμε να βρούμε κριμένες ιδιότητες χωρίς όποιες ετικέτες στα δεδομένα. Δεν υπάρχει σωστή και λάθος απάντηση .Ένα παράδειγμα unsupervised learning είναι οι clustering αλγόριθμοι.

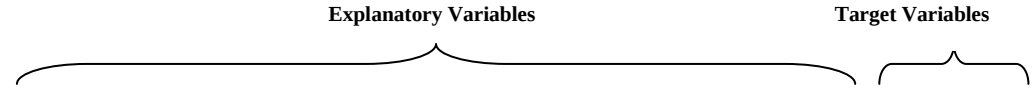
4.2 Classification in Data Analytics

Το να κατατάσσουμε (classify) τα αντικείμενα σε κατηγορίες είναι κάτι που κάναμε από την παιδική μας ηλικία όταν προσπαθούσαμε να κατατάξουμε τα δέντρα σε εσπεριδοειδή φυλλοβόλα και άλλα. Για να το πετύχουμε αυτό δίναμε σε κάθε κατηγορία κάποια χαρακτηριστικά και με βάση αυτά κατατάσσαμε το κάθε δέντρο στην σωστή κατηγορία.

Για να το αυτοματοποιήσουμε αυτό με ηλεκτρονικούς υπολογιστές χρησιμοποιούμε αλγόριθμους machine learning. Ένα παράδειγμα είδη υπάρχοντος προβλήματος που χρησιμοποιεί classification είναι τα antiviruses . Το antiviruses δεν είναι τίποτα άλλο από ένα εκπαιδευμένο εργαλείο το οποίο αναλύει το binary code κάποιου εκτελέσιμου αρχείου και προσπαθεί να καταλάβει αν ανήκει σε μια από τις κατηγορίες των κακόβουλων λογισμικών που έχει στην βάση δεδομένων του. Η ικανότητα να εντοπίζει σωστά αν ένα εκτελέσιμο είναι κακόβουλο η όχι και τι είδους κακόβουλο λογισμικό είναι, κατατάσσει ένα antivirus σε αποδοτικό η όχι.

Το Classification είναι Supervised leaning technique και χρησιμοποιείται αρκετά για πρόβλεψη για παράδειγμα εντοπισμός των spam emails. Στη διαδικασία της κατηγοριοποίησης, μέσα από το σύνολο των δεδομένων που μας δίνεται προσπαθούμε να εξακριβώσουμε την ταυτότητα του κάθε αντικειμένου. Για να το πετύχουμε αυτό ψάχνουμε για στοιχεία μέσα στα δεδομένα που να μας δίνουν τις κατάλληλες πληροφορίες. Αυτά τα στοιχεία ονομάζονται Explanatory Variables ή διαφορετικά Features και οι τελικές κατηγορίες για τις οποίες ενδιαφερόμαστε τα ονομάζουμε Target Variables ή Labels.

Έστω τώρα ότι θέλουμε να δημιουργήσουμε ένα μοντέλο του οποίου θα του δίνουμε κάποια χαρακτηριστικά ζώων και θα μας λέει αν είναι θηλαστικό, ερπετά , πουλιά και πολλά άλλα(Target Variables). Παράδειγμα χαρακτηριστικών μπορεί να είναι ο τρόπος που γεννάνε, αν έχουν ουρά , αν έχουν λέπια, πόσα πόδια έχουν.



Πίνακας 4.1

Need Air	Legs Numb	Have Scales	Lay Eggs	Have Wings	Type
Yes	**4**	**No**	**No**	**No**	**Mammal**
Yes	**2**	**No**	**Yes**	**Yes**	**Birds**
No	**0**	**Yes**	**Yes**	**No**	**Fish**
Yes	**4/0**	**Yes**	**Yes**	**No**	**Reptiles**

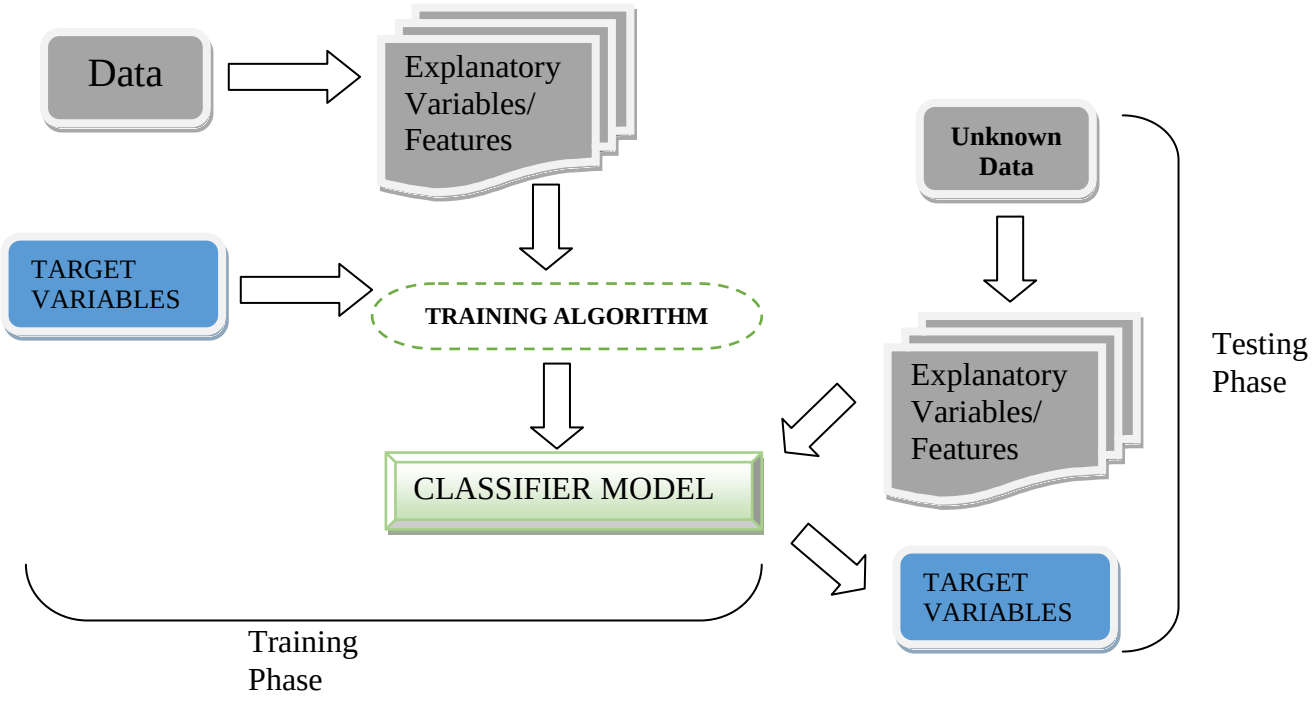
Στο classification έχουμε την δυνατότητα να ελέγξουμε αν το μοντέλο μας με τον αλγόριθμο που χρησιμοποιήσαμε και τα δεδομένα που του δώσαμε μας δίνει καλά αποτελέσματα. Για να το πετύχουμε αυτό τα δεδομένα που έχουμε στην κατοχή μας τα σπάζουμε σε 2 κατηγόριες:

- **Training Dataset:** Είναι τα δεδομένα τα οποία έχουμε στην κατοχή μας για να μπορέσουμε να εκπαιδεύσουμε τον classifier μοντέλο μας. Συνήθως χρησιμοποιούμε το 70% του υπάρχοντος συνόλου δεδομένων.

- **Test Data Set:** Είναι το υπόλοιπο 30% και χρησιμοποιείτε μετά το training dataset για να επαληθεύσουμε το μοντέλο μας. Αυτό γίνεται για να δούμε αν η εκπαίδευση μας είναι καλή η όχι αφού θα έχουμε δεδομένα τα οποία ξέρουμε σε πια κατηγορία ανήκουν άρα μπορούμε να δούμε πόσο καλά έγινε η κατανομή.

Η διαδικασία που ακολουθείται για να κτίσουμε τον Classifier:

1. Συγκέντρωση όλων των δεδομένων που έχουμε και μπορούν να μας βοηθήσουν
2. Μετατροπή δεδομένων στη μορφή που θέλουμε καθώς και διαγραφή των καινών και μη χρήσιμων δεδομένων.
3. Προσδιορισμός των Target Variables
4. Προσδιορισμός των Explanatory Variables /Features
5. Εύρεση Αλγόριθμου Εκπαίδευσης του μοντέλου μας
6. Εκπαίδευση του μοντέλου χρησιμοποιώντας το Training Dataset
7. Αξιολόγηση με το Testing Dataset



Overfitting and Underfitting

Στη διαδικασία εκπαίδευσης του μοντέλου μας, πρέπει να προσέξουμε την ύπαρξη των outliers. Outliers είναι τα αντικείμενα που έχουμε στα δεδομένα εισόδου μας τα οποία δεν έχουν τα χαρακτηριστικά που θα έπρεπε να έχουν για την κατηγορία που ανήκουν. Αν λάβουμε υπόψη μας αυτά τα δεδομένα ίσως να μας δώσουν λάθος αποτελέσματα.
Για την αντιμετώπιση αυτού του προβλήματος χρησιμοποιούμε fuzzy-logic-based algorithms και Distance-based techniques.

Το *overfitting* εμφανίζεται χρησιμοποιούμε όλα τα δεδομένα για να κτίσουμε το μοντέλο μας χρησιμοποιώντας καθαρή απομνημόνευση. Αντί να βρίσκουμε ένα γενικό μοτίβο, το μοντέλο απλά το απομνημονεύει. Συνήθως όταν έχουμε overfitting το μοντέλο είναι πιο περίπλοκο και είναι επιρρεπή σε λάθος αποφάσεις. Κάποιες μέθοδοι αντιμετώπισης του προβλήματος είναι η μείωση των features και το regularization κάποιων από αυτών. Κάτι άλλο που συνηθίζεται είναι να εκπαιδεύουμε το μοντέλο μας με ένα μέρος των δεδομένων που έχουμε στην κατοχή μας και να το επαληθεύουμε με το υπόλοιπο. Αυτή η μέθοδος ονομάζεται cross-validation.

Το *underfitting (high bias)* εμφανίζεται όταν ο αλγόριθμος δεν εκπαιδεύεται επαρκώς και δεν μπορεί να δημιουργήσει μοτίβα. Σε αυτό το πρόβλημα δεν θα βοηθήσει αν δώσουμε περισσότερα δεδομένα αυτό που χρειάζεται είναι περισσότερα explanatory variables.

4.3 Naïve-Bayes Θεώρημα

Υπό προϋπόθεση πιθανότητα (Conditional Probability):

Είναι η πιθανότητα να συμβεί κάτι δεδομένου ότι κάτι άλλο έχει είδη συμβεί. Συμβολίζεται ως P(A/B) και διαβάζεται ως η πιθανότητα του A δεδομένου του B.
Μαθηματικά **P(A/B) = P(A ∩ B) / P(B).**
Ένα παράδειγμα είναι η πιθανότητα μέσα από ένα κουτί με δύο άσπρες και δύο μαύρες μπάλες να τραβήξω άσπρη θεωρώντας ότι την αμέσως προηγούμενη φορά τράβηξα μαύρη.

Bayes theorem:

Μας δίνει την σχέση μεταξύ δύο υπό προϋπόθεση πιθανοτήτων.
P (A/B) = P (B/A) P (A) / P (B).

4.4 Naive Bayes Algorithm

Όπως είδαμε στο Bayes θεώρημα το αποτέλεσμα μας εξαρτάται μόνο από μια προϋπόθεση (δεδομένου του B) . Αντιθέτως όμως στο πρόβλημα του classification που έχουμε να λύσουμε οι προϋποθέσεις είναι πάρα πολλές και πρέπει με βάση αυτές να μπορούμε να εκπαιδεύσουμε το μοντέλο μας.

Ο Naïve Bayes αλγόριθμος αποσυνδέει τις πολλαπλές προϋποθέσεις και αντιμετωπίζει την κάθε μία ανεξάρτητα από τις άλλες.

P (outcome| multiple evidence) =
P (Evidence 1 | outcome) * P (evidence 2 |outcome)…. /P (Evidence)

Αυτό που πρέπει να κάνουμε είναι να βρούμε τις προϋποθέσεις (features) για να αποφασίσουμε πόσο πιθανόν είναι το outcome να ανήκει σε μια τάξη.

Παράδειγμα:

Έστω ότι έχουμε 1000 κομμάτια φρούτα και ξέρουμε κάποια χαρακτηριστικά γι' αυτά. Για παράδειγμα το χρώμα τους , το μέγεθος τους έχουν και την γεύση τους.

Πίνακας 4.2

Τύπος	Γεύση γλυκιά	Γεύση μη γλυκιά	Χρώμα Κίτρινο	Χρώμα μη Κίτρ.	Μέγεθος Μεγάλο	Μέγεθος Μικρό	Σύνολο
Μπανάνα	350	150	450	50	400	100	500
Μήλο	150	150	100	200	0	300	300
Άλλο	150	50	50	150	100	100	200
Σύνολο	650	350	600	400	500	500	1000

Ερώτημα:
Η πιθανότητα να έχω μήλο

Λύση:
P (Μήλο) = Αριθμός Μήλων/ Συνολικά φρούτα = 300/1000= 0.3

Πιθανότητα τώρα για κάθε ένα από τα χαρακτηριστικά(features):

P(Γλυκιά γεύση) = 650 / 1000 = 0.65
P(Κίτρινα) = 600/1000 =0.6
P(Μεγάλα σε μέγεθος)=500/1000=0.5

Επομένως τώρα αν θέλομε να δούμε τι φρούτο θα έχουμε αν είναι

- κίτρινο
- μεγάλο σε μέγεθος
- γλυκό.

P(Μήλο| γλυκό, μεγάλο, κίτρινο) = P(γλυκό | Μήλο) * P(μεγάλο | Μήλο) * P(κίτρινο| Μήλο) * P(Μήλο) / P(γλυκό) * P(μεγάλο) * P(κίτρινο)

P(Μπανάνα| γλυκό, μεγάλο, κίτρινο) = P(γλυκό | Μπανάνα) * P(μεγάλο | Μπανάνα) * P(κίτρινο| Μπανάνα) * P(Μπανάνα) / P(γλυκό) * P(μεγάλο) * P(κίτρινο)

…

Και ούτω κάθε εξής. Τέλος αφού υπολογίσουμε όλες τις υπό προϋπόθεση πιθανότητες , επιλέγουμε την πιο μεγάλη από αυτές.

4.5 Naïve Bayes in Text classification

Όταν έχουμε για είσοδο ένα text αρχείο , πρέπει μέσα από τις λέξεις να βρούμε τις explanatory και target μεταβλητές βάση των οποίων θα δημιουργήσουμε ένα διάνυσα.

Bag of Words:

Είναι ο προσδιορισμός ενός αντικειμένου, μιας κατηγορίας ως σύνολο από λέξεις. Όπως όλα τα σύνολα δεν λαμβάνεται υπόψη η σειρά των λέξεων , γραμματική και σημεία στίξης. Ως feature παίρνουμε την κάθε λέξη ξεχωριστά.

Term frequency:

Αυτό μετρά το πόσες φορές παρουσιάστηκε η λέξη. Επομένως η σημαντικότητα της λέξης αυξάνεται με τον αριθμό των εμφανίσεων της στο αρχείο . Στο τέλος θα καταλήξουμε σε ένα πίνακα που ονομάζεται **term of frequency table**.

Για να μπορέσουμε να καταλήξουμε σε αυτό τον πίνακα εφαρμόζονται οι πιο κάτω τεχνικές:

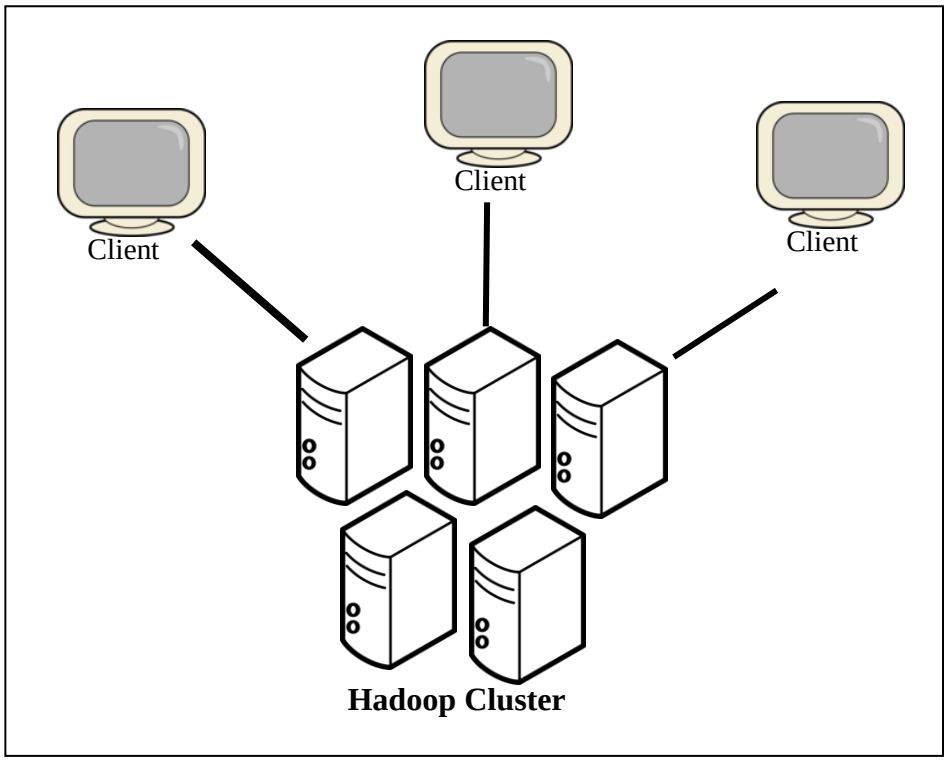
- **Streaming of words:** Λέξεις που έχουν διαφορετικές καταλήξεις αλλά ίδια βάση, διαγράφονται οι καταλήξεις και η λέξεις γίνονται ίδιες για παράδειγμα η λέξη «Υπολογίζω» «Υπολογιζόμενα» θα γίνει «Υπολογιζ»
- **Case Normalization:** Όλες οι λέξεις γίνονται στο lowercase
- **Stop Word removal:** Υπάρχουν κάποιες λέξεις που εμφανίζονται σχεδόν σε κάθε αρχείο. Αυτές οι λέξεις τις ονομάζουμε Stop Words. Κατά τη διάρκεια της εξαγωγής των σημαντικών features αυτές τις λέξεις τις αγνοούμε γιατί δεν θα είναι βοηθητικές αργότερα στον υπολογισμό της πιθανότητας. Παράδειγμα τέτοιων λέξεων είναι οι «το, τα , είναι, αυτό κ.α.»
- **Inverse document frequency:** Αν μια λέξη εμφανίζεται σε πολλά κείμενα πολλών κατηγοριών αυτό σημαίνει πως η εμφάνιση της δεν δίνει σίγουρα αποτελέσματα για το σε πια κατηγορία ανήκει το κείμενο. Αν όμως αυτή η λέξη εμφανίζεται μόνο σε μια κατηγορία η πιθανότητα το κείμενο να βρίσκεται στην συγκεκριμένη κατηγορία όταν βρούμε αυτή τη λέξη πρέπει να είναι μεγαλύτερη. Με λίγα λόγια η πιθανότητα δεδομένης μιας λέξης να έχουμε μια κατηγορία είναι αντίθετου του αριθμού εμφανίσεων.
  IDF(t)= 1 + log (total number of documents / number of documents containing t)

Term frequency and inverse term frequency: TFIDF(t,d)= TF(t, d) * IDF (t).

5.1 Επισκόπηση Hadoop

Το Hadoop είναι ένα ανοιχτού πηγαίου κώδικα framework που δημιουργήθηκε από την Apache το 2003 . Είναι ένα λογισμικό το οποίο επιτρέπει την επεξεργασία μεγάλων δεδομένων σε κατανεμημένο σύστημα. Τα κύρια χαρακτηριστικά του Hadoop είναι τα ακόλουθα:

- **Προσβάσημο (Accessible):** Μπορεί να τρέξει πάνω σε cluster από παραδοσιακούς υπολογιστές ή σε μηχανές που βρίσκονται στο cloud.
- **Ανθεκτικό(Robust):** Λόγω του ότι προοριζόταν να τρέχει σε παραδοσιακούς υπολογιστές , χτίστηκε έχοντας κατά νου τις συχνές βλάβες που προκαλείται στο λογισμικό. Επομένως μπορούμε να πούμε ότι διαχειρίζεται πολύ καλά τις βλάβες που εμφανίζονται.
- **Επεκτάσιμο (Scalable):** Διαχειρίζεται την αύξηση του μεγέθους των δεδομένων γραμμικά λόγω του κατανεμημένου χαρακτήρα του.
- **Απλό (Simple):** Επιτρέπει στον χρήστη να γράφει γρήγορα και αποδοτικά παράλληλα προγράμματα.

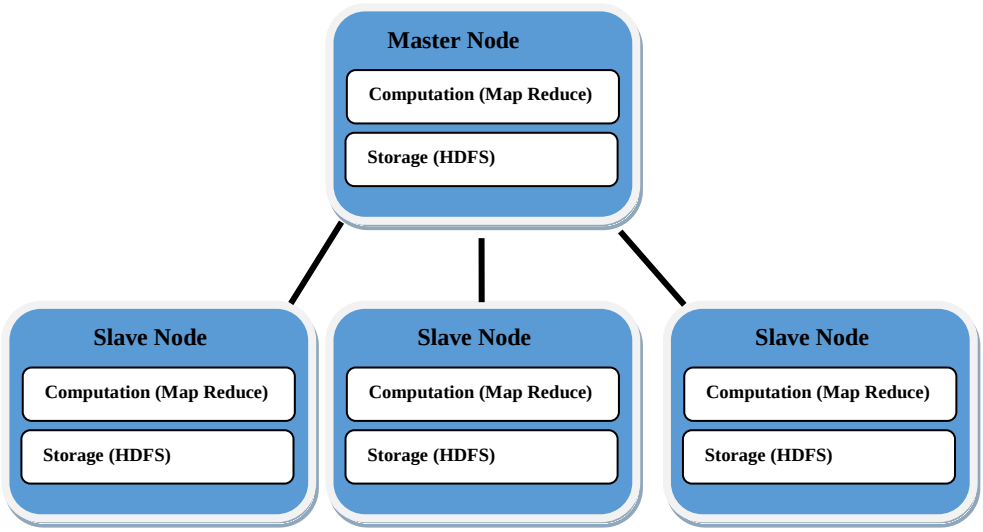


Σχήμα 5.1 Το Hadoop Cluster αποτελείται από πολλά παράλληλα μηχανήματα τα οποία αποθηκεύουν και επεξεργάζονται μεγάλο αριθμό δεδομένων εισόδου. Εδώ βλέπουμε πως οι clients αιτούνται jobs στο cluster δίνοντας τα δεδομένα εισόδου και παίρνουν πίσω το αποτέλεσμα.

Όπως βλέπουμε στο σχήμα και η υπολογιστική αλλά και η αποθηκευτική μονάδα ανήκουν στο Hadoop Cluster. Ο χρήστης στέλνει μόνο τα MapReduce προγράμματα που πρέπει να εκτελεστούν τα οποία είναι συνήθως μικρού μεγέθους.

5.2 Αρχιτεκτονική Hadoop Cluster

Συγκεκριμένα το Hadoop είναι Master – Slave Distributed αρχιτεκτονική που αποτελείται από την μονάδα αποθήκευσης (HDFS – Hadoop Distributed File System) και την μονάδα επεξεργασίας (Map Reduce)



Σχήμα 5.2

MapReduce master:

Είναι υπεύθυνος για να οργανώνει σε ποιους nodes θα γίνεται ο κάθε υπολογισμός

HDFS master:

Είναι υπεύθυνος για να μεριάζει τα δεδομένα στους slave και να ξέρει ποιος έχει τι ανά πάσα στιγμή.

5.3 Daemons

Τα σημαντικότερα κομμάτια του Hadoop αποτελούν οι daemons, που είναι υπεύθυνοι για κάθε λειτουργία που συμβαίνει. Συγκεκριμένα έχουμε :

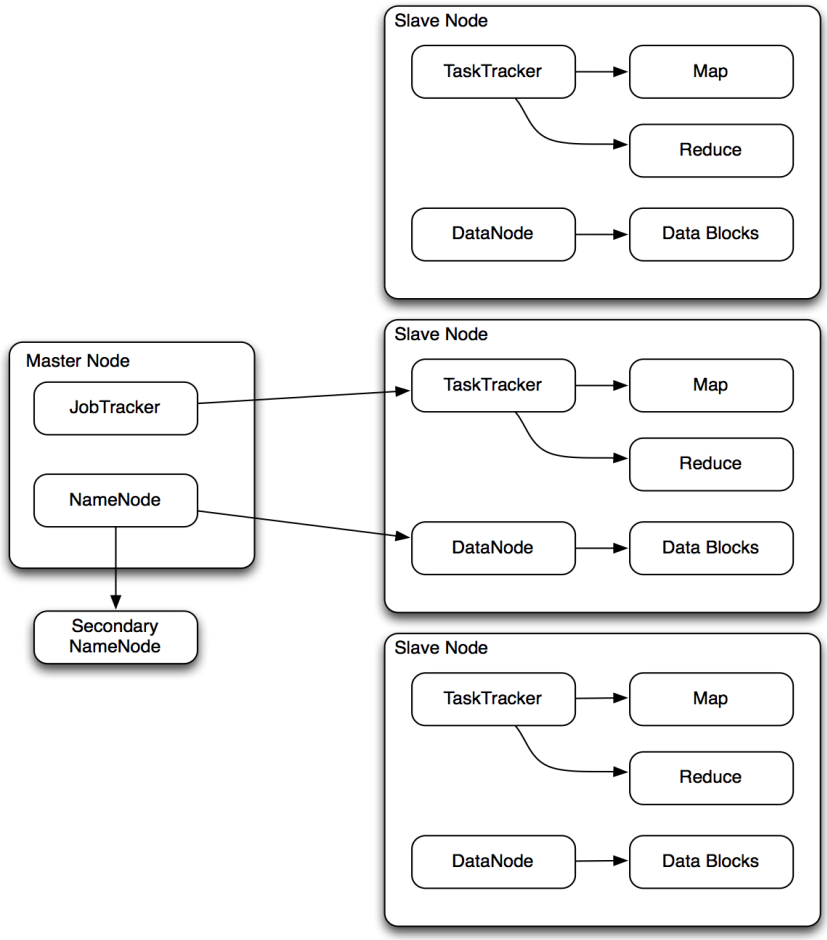
Τον **Namenode**, είναι ο master node ο οποίος φιλάει μεταπληροφορίες για όλα τα αρχεία που βρίσκονται στο cluster.

Για λόγους fault tolerance το mahout έχει ακόμα ένα daemon, τον **SecondaryNameNode** ο οποίος βρίσκεται εκεί για να παίρνει περιοδικά τις πληροφορίες που έχει ο Namenode έτσι ώστε αν σε περίπτωση που πάθει κάτι ο node πάνω στον οποίο βρίσκεται να μην καταρρεύσει όλο το σύστημα.

Ο **Datanode** είναι οι slaves οι οποίοι είναι υπεύθυνοι για τα δεδομένα του node , να γράφου και να διαβάζουν τοπικά καθώς και να ενημερώνουν το Namenode.

Στην άλλη μεριά της υπολογιστικής μονάδας του Hadoop έχουμε άλλους δυο daemons. Ο **Jobtracker** κατοικεί στον master node και είναι υπεύθυνος να λαμβάνει την εργασία από τον χρήστη και να την διαμοιράζει στο σύστημα .

Οι αρμόδιοι για κάθε κόμβο είναι οι **TaskTracke**r ο οποίοι είναι υπεύθυνοι για να τρέχουν τις map reduce συναρτήσεις.



Σχήμα 5.3

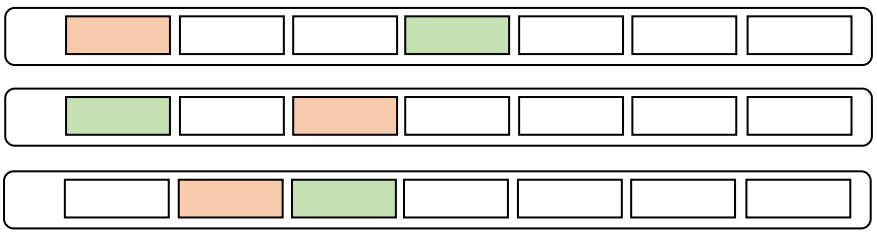
5.3 Χαρακτηριστικά HDFS

Το HDFS ή καλύτερα το Hadoop Distributed File System είναι η αποθηκευτική μονάδα του Hadoop Cluster. Όπως λέει και το όνομα του είναι ένα κατανεμημένο File System το οποίο μοντελοποιήθηκε με βάση το Google File System (GFS) που δημοσιεύτηκε το 2003[6]. Όπως είδαμε πιο πάνω, το Hadoop έχει κάποια συγκεκριμένα χαρακτηριστικά λειτουργίας κάποια από τα οποία απορρέουν από τις πιο κάτω ιδιότητες του HDFS:

1. Είναι ικανό να αποθηκεύει και να διαχειρίζεται μεγαλύτερα block sizes (π.χ 100 TB) σε σύγκριση πάντα με άλλα file systems που θα αδυνατούσαν. Τα αρχεία στο HDFS αποθηκεύονται σε 1 ή περισσότερα blocks και κάθε block είναι τυπικά 64MB ή μεγαλύτερο.
2. Χρησιμοποιεί data locality τεχνικές για να μειώσει τις καθυστερήσεις που προκύπτουν από I/O

Αυτά έχουν σαν αποτέλεσμα το HDFS να μπορεί να υποστηρίξει υψηλό throughput και να μειώνει τις καθυστερήσεις του disk seek.

3. Data Replication: Τα blocks στα οποία αποθηκεύονται τα αρχεία αντιγράφονται σε πολλούς host στο cluster για σκοπούς πρόληψης σφαλμάτων (Figure 5.3).



Σχήμα 5.3

4. Είναι CheckSummed File System: Κάθε block έχει ένα αντίστοιχο checksum το οποίο ελέγχεται για να διαπιστωθεί αν το block είναι άρτιο. Αν δεν είναι, στέλνεται η πληροφορία στο HDFS master ο οποίος αναλαμβάνει να αντιγράψει το συγκεκριμένο block καθώς και την διαγραφή του είδη υπάρχοντος.

Αυτά δίνουν στο HDFS επεκτασιμότητα και διαθεσιμότητα(Scalability and Availability)

Αξίζει να σημειωθεί πως το HDFS δεν είναι native Unix FileSystem και κατά συνέπεια εντολές όπως η ls , cp , cat δεν μπορούν να δουλέψουν αλλά ούτε μπορεί να εκτελέσει τα κανονικά read/write operations όπως η fopen(). Παρόλα αυτά το Hadoop παρέχει ένα σύνολο από εντολές που δουλεύουν παρόμοια με το Unix για να μπορεί ο χρήστης να πλοηγείτε εύκολα.

5.4 Πως γράφουμε σε HDFS

Τα στάδια που ακολουθούνται είναι τα ακόλουθα:

1. Φορτώνει τα core-default.xml και core-site.xml όπου και ορίζεται το FileSystem

2. Δημιουργούμε stream για το file στο file system αφού έχει γίνει η διαδικασία από τον NameNode για αν αποφασιστεί πια DataNodes θα χρησιμοποιηθούν για να γράψουν το block.

3. Αντιγραφή του περιεχομένου από το τοπικό αρχείο στο αρχείο στο HDFS. Αφού κάθε block έχει ολοκληρώσει την αντιγραφή του ο NameNode επαναλαμβάνει την διαδικασία για να επιλέξει τους επόμενους Datanodes που θα αναλάβουν το block που ακολουθεί.

<u>Στάδιο 1:</u> Ορισμός File System

Το Hadoop πρώτα απ' όλα ορίζει πιο θα είναι το σύστημα το οποίο θα χρησιμοποιήσει διαβάζοντας την τιμή fs.default.name από το core-default.xml και core-site.xml.(Το core-site.xml κάνει overwrite κάποιες τιμές του core-default.xml που επιθυμεί ο χρήστης)
Στην περίπτωση που θα φορτωθεί το HDFS file system η τιμή του fs.default.name θα είναι κάτι παρόμοιο με το hdfs://namenode:9000.

<u>Στάδιο 2:</u> Σύνδεση με τον Namenode για να αποφασιστεί που θα γραφτεί το block.

Μια Hadoop κλάση , η DistributedFileSystem δημιουργεί την *DFSClient* η οποία διαχειρίζεται οποιαδήποτε επικοινωνία με τον NameNode και τους DataNodes. Συγκεκριμένα διαβάζει ένα αριθμό από configuration files μέσα στα οποία συμπεριλαμβάνεται το size του block (dfs.block.size) και το replication factor(dfs.replication). Τέλος δημιουργεί μία σύνδεση του client με τον Namenode όπως φαίνεται στο figure…

Όπως προαναφέραμε το HDFS φιλάει τα blocks σε ένα ή περισσότερα blocks. Επομένως, για κάθε block ο namenode καλείται να αποφασίσει ποιος Datanode θα φιλάει ποια blocks. Εδώ ακολουθείται και η τεχνική του Data Locality που είχαμε πει και πιο πάνω. Πρώτα προσπαθεί να φυλάξει τα δεδομένα στο τοπικό node αν ο client τρέχει σε DataNode.

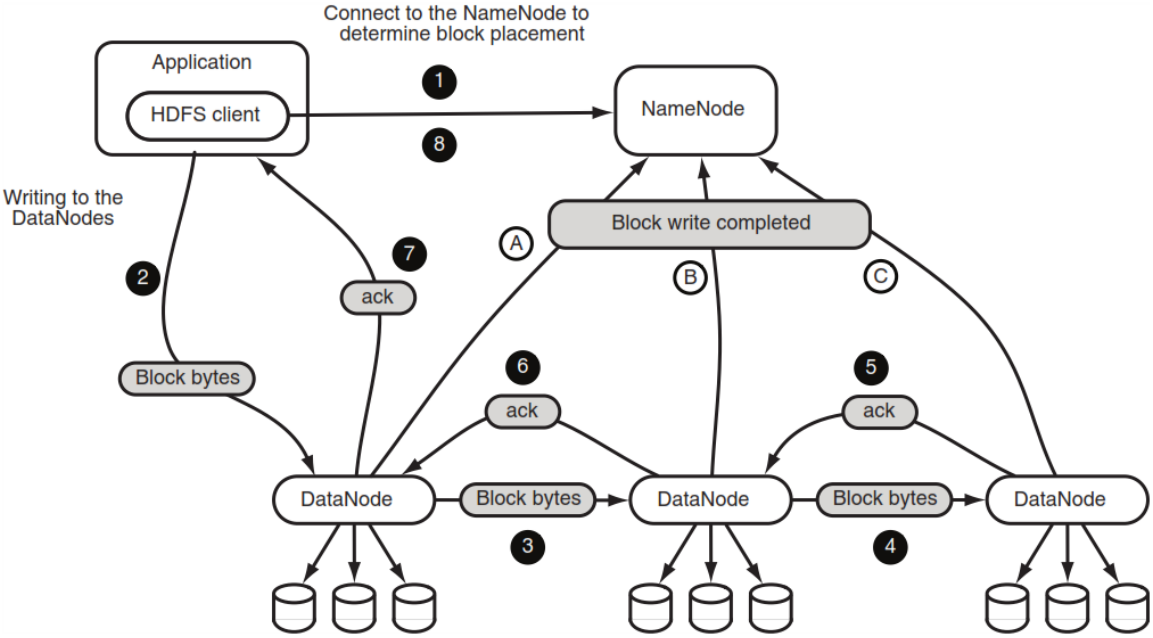
<u>Οι Nodes που θα επιλεχτούν κατατάσσονται με την εξής σειρά:</u>

i. **Τοπικός δίσκος:** Network I/O είναι ακριβό επομένως διαβάζοντας από τον τοπικό δίσκο μηδενίζει οποιαδήποτε Network I/O καθυστέρηση

ii. **Τοπικό Rack:** Ο λόγος είναι επειδή συχνά οι ταχύτητες σε υπολογιστές μέσα στο ίδιο Rack είναι ψηλότερες παρά έξω από αυτό.

iii. Άλλο

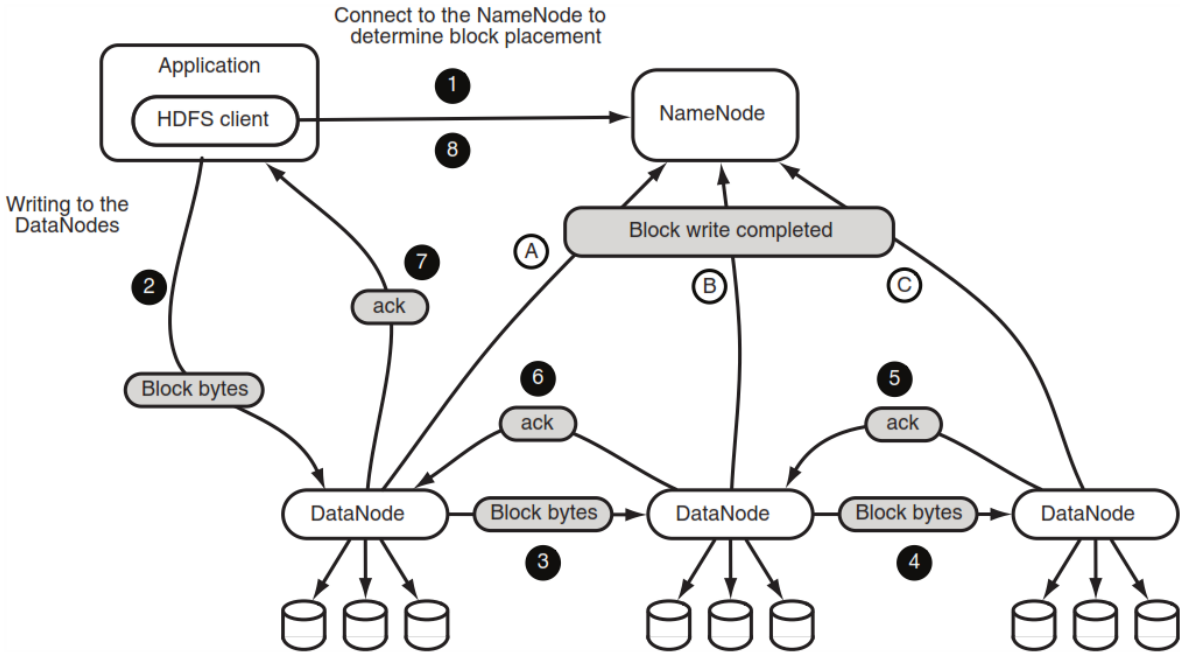
Σε ένα rack-aware σύστημα ο NameNode θα φροντίσει τουλάχιστον ένα αντίτυπο του block βρίσκεται σε άλλο rack.
Ακολούθως μια λίστα από τα DataNodes για το κάθε block επιστρέφεται στον client με την σειρά που αναφέραμε πιο πάνω. Αφού έχουμε πολλά blocks σε πολλά nodes αυτό σημαίνει πως όταν γράφουμε πρέπει να τα ενημερώνουμε όλα.



Σχεδιάγραμμα 5.4 Ροή δεδομένων στο HDFS

Στάδιο 3: Γράφοντας τα blocks στον επιλεγόμενο Datanode

Το HDFS χρησιμοποιεί pipelining για να επιτύχει το γράψιμο του block σε διάφορους Datanodes. Αφού ο Namenode έχει πιάσει την λίστα με τους Datanodes ο χρήστης στέλνει το πρώτο block από δεδομένα στο πρώτο DataNode(Βήμα 2 στο figure) όπου προωθεί το block στον επόμενο DataNode και ούτω κάθε εξής (Βήμα 3 στο figure) μέχρι το block να φτάσει και στον τελευταίο (Βήμα 4 στο figure) το οποίο αφού παραλάβει το block στέλνει acknowledgment με pipelining μέθοδο ξανά αλλά τώρα από την αντίθετη κατεύθυνση μέχρι να φτάσει στον client(Βήματα 5,6,7 στο figure…). Ο χρήστης τώρα το μόνο που έχει να κάνει είναι να συνεχίσει γράφοντας το επόμενο block. Όταν κάθε DataNode γράψει το αρχείο τοπικά τα blocks μεταφέρονται από την προσωρινή μνήμη στην μόνιμη και κάθε Datanode ειδοποιεί τον Namenode για το block το οποίο φύλαξαν (Βήματα A, B και C στο figure..)



Σχήμα 5.5 Σχεδιάγραμμα ροής της HDFS write λειτουργίας.

5.5 Διαδικασία διαβάσματος από HDFS

Τα στάδια που ακολουθούνται είναι τα ακόλουθα:

1. Δημιουργούμε stream για το file στο HDFS.Αυτό συνεπάγεται αίτημα στον Namenode πως θέλουμε να διαβάσουμε το συγκεκριμένο αρχείο και μας επιστρέφει πίσω τις σχετικές πληροφορίες (Datanode)για τα δέκα πρώτα blocks του αρχείου.

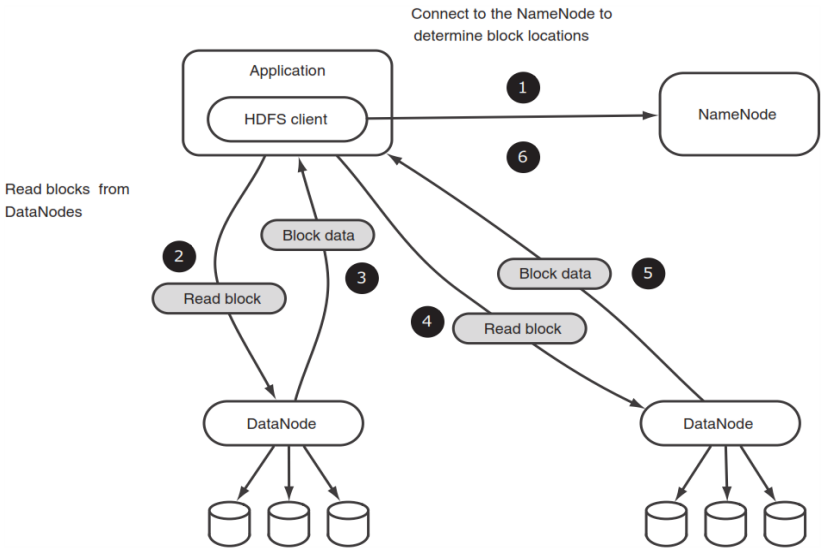
2. Διαβάζουμε το περιεχόμενο του αρχείου και το γράφουμε στο standard output. Μετά τα δέκα πρώτα blocks που διαβάσαμε ξανά επικοινωνούμε με τον Namenode ο οποίος θα μας στείλει τα επόμενα δέκα blocks.

Στάδιο 1: Σύνδεση με τον Namenode για να αποφασιστεί που βρίσκεται το κάθε block.

Όπως και προηγουμένως η DFSClient κλάση είναι υπεύθυνη για την επικοινωνία μεταξύ του Namenode και των Datanodes. Μόλις ο client αιτηθεί να διαβάσει ένα υπάρχον αρχείο ο DFSClient δημιουργεί ένα DFSInputStream το οποίο ρωτάει τον Namenode μεταπληροφορίες για τα 10 πρώτα blocks(Βήμα 1 στο figure).Για κάθε block, ο Namenode ταξινομεί τους DataNodes με βάση πια από αυτά βρίσκονται πιο κοντά στον client. Όπως και πριν με το γράψιμο στο HDFS προτεραιότητα έχει ο local Datanode(Στο ίδιο φυσικό node) , στη συνέχεια Datanodes μέσα στο ίδιο Rack με τον client node και τέλος τα Datanodes που βρίσκονται σε διαφορετικά Racks.

<u>Στάδιο 2:</u> Διάβασμα των block από τους DataNodes

Όταν ο client παραλάβει την λίστα με τα blocks από τον Namenode παίρνει τον πρώτο Datanode από την λίστα και δημιουργεί ένα stream με αυτόν.(Βήμα 2 στο figure) και αρχίζουν να διαβάζουν bytes μέχρι να φτάσουν στο τέλος του αρχείου όπου και κλίνει η σύνδεση. Μετά το τέλος του διαβάσματος του πρώτου block επαναλαμβάνει την ίδια διαδικασία για το δεύτερο block και Datanode (Βήμα 3 ,4 στο figure..). Αν το αρχείο είναι μεγαλύτερο από 10 blocks θα επαναλάβει το Στάδιο 1.



Σχήμα 5.6 Διάβασμα αρχείου από το HDFS

Αν τα δεδομένα φυλάσσονταν σε μόνο μια κεντρική μονάδα τότε το bottleneck θα υπήρχε στο bandwidth του server. Με την προσθήκη περισσότερων μηχανών για επεξεργασία μόνο μας βοηθά μόνο ενώσω δεν χρησιμοποιούμε την μονάδα αποθήκευσης.

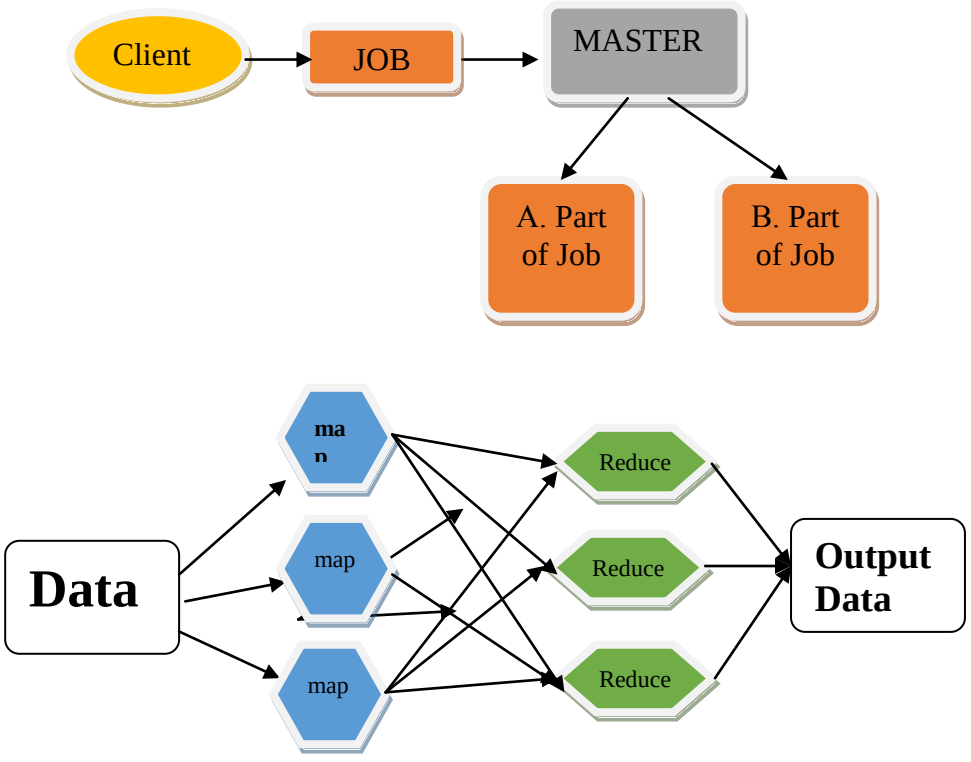
5.6 Map Reduce

Είναι ένα κατανεμημένο σύστημα επεξεργασίας batch δεδομένων το οποίο υλοποιήθηκε μετά το άρθρο της Google στο Map Reduce [5].
Επιτρέπει την παράλληλη επεξεργασία μεγάλου αριθμού δεδομένων μέσα σε κάποια λεπτά, πράγμα που οι συμβατικές σειριακές προγραμματιστικές μέθοδοι θα χρειαζόταν μέρες για να ολοκληρώσουν.

Το μοντέλο του MapReduce απλοποιεί την παράλληλη επεξεργασία απομακρύνοντας τον χρήστη από πολύπλοκες έννοιες και εργασίες που αφορούν την επεξεργασία σε κατανεμημένα συστήματα. Με αυτή την αφαιρετικότητα δίνει στον χρήστη την ευκαιρία να ασχολείται με την πραγματική ουσία των αποτελεσμάτων του και των δεδομένων παρά με άλλα θέματα υλικού.

Τα MapReduce προγράμματα εκτελούνται σε δύο βασικές φάσεις που ονομάζονται mapping και Reducing. Για κάθε φάση ορίζεται από τον χρήστη μια συνάρτηση που ονομάζονται mapper και reducer αντίστοιχα.
Η υποβαλλόμενη δουλεία από την χρήστη μοιράζεται στους παραλληλοποιημένους maps workers και ακολούθως στους reduce workers. Πιο συγκεκριμένα οι map works παίρνουν το αρχείο εισόδου που τους δίνει ο χρήστης το επεξεργάζονται και παράγουν ως έξοδο ένα key/values αρχείο το οποίο παίρνουν σαν είσοδο οι reduce workers.

Όταν τελειώσει η επεξεργασία που γίνεται από τους mappers δημιουργούμε στο δίσκο(αν τα δεδομένα μας είναι πολύ μεγάλα που υπερβαίνουν την κύρια μνήμη) αρχεία τα οποία χωρίζονται μεταξύ τους με βάση το key, με άλλα λόγια όλα τα values με το ίδιο key φυλάσσονται στο ίδιο αρχείο και στην συνέχεια στέλνονται στον reducer ο οποίος θα αναλάβει την μετέπειτα επεξεργασία όλων των τιμών με το συγκεκριμένο key. Όπως καταλαβαίνουμε σε περίπτωση load balance (Όταν ένας mapper έχει περισσότερη δουλειά από τους άλλους) όλοι οι reducers θα περιμένουν μέχρι να τελειώσει αυτός.

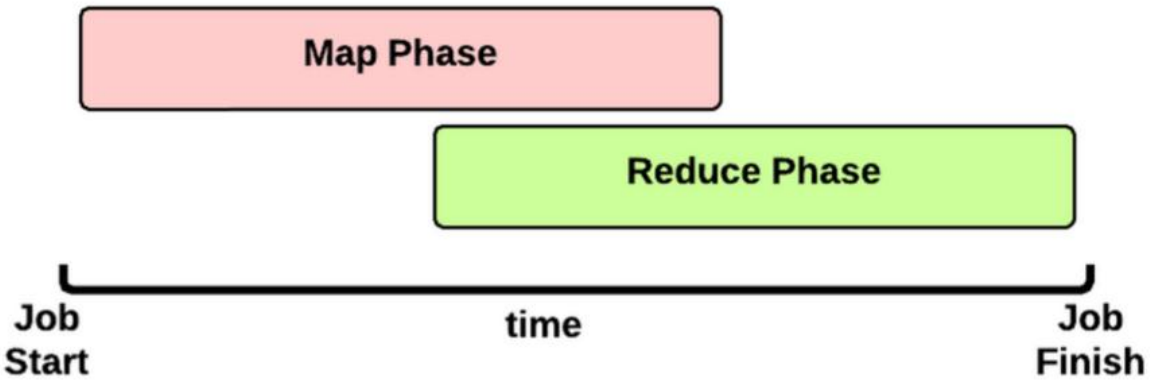


Σχεδιάγραμμα εισαγωγής εργασίας MapReduce από τον χρήστη

5.7 Anatomy MapReduce Job

MapReduce job

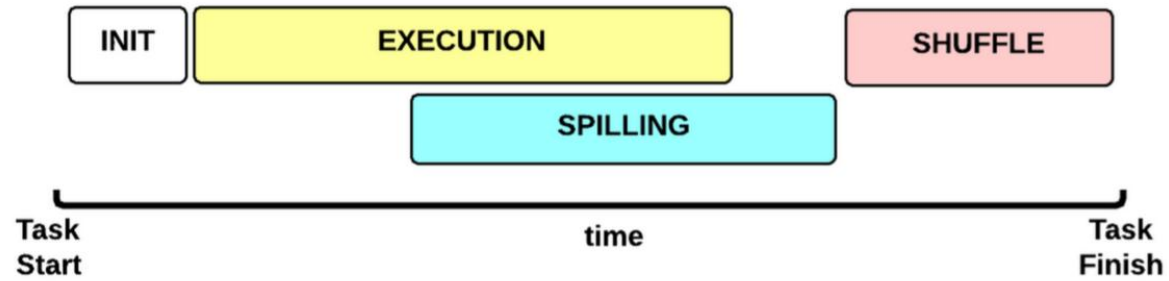
Ονομάζεται η εργασία κατά την οποία ο mapper παίρνει τα δεδομένα τα επεξεργάζεται και τα δίνει στον reducer ο οποίος τρέχει μια άλλη συνάρτηση πάνω στα δεδομένα τα οποία εξήγαγε ο mapper. Μπορούμε να έχουμε ένα MapReduce job ή πολλά συνδεδεμένα.



Σχήμα 5.7 Χρονοδιάγραμμα MapReduce [9]

Στο πιο πάνω σχεδιάγραμμα φαίνονται οι δύο φάσεις μιας Map Reduce εργασίας .Στη Map φάση εκτελείται η Map συνάρτηση και στη Reduce φάση εκτελείται η Reduce συνάρτηση.

Στα ακόλουθα διαγράμματα περιγράφεται η κάθε φάση ξεχωριστά.

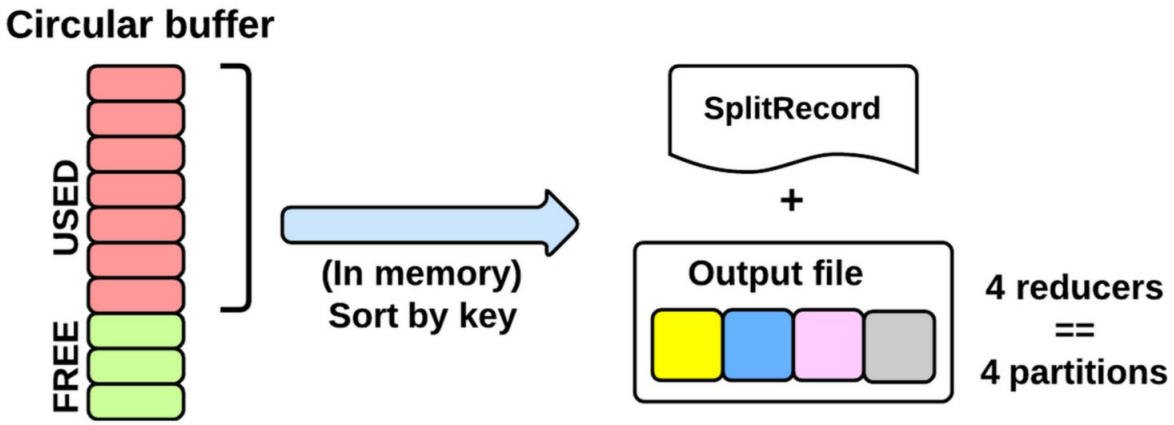


Σχήμα 5.8 Χρονοδιάγραμμα Map Job[9]

Φάσεις του Map Job:

INIT:
Δημιουργία Map αντικειμένου, προσδιορισμός της εργασίας και των δεδομένων εισόδου.

EXECUTION:
Στη συγκεκριμένη φάση ο κάθε mapper σπάζει το κομμάτι των δεδομένων που του έχει δοθεί σε μικρότερα (αν είναι απαραίτητο) και στην συνέχεια το χωρίζει σε tuples της μορφής <key,value>΄όπου key ο αριθμός της γραμμής και value το κείμενο της συγκεκριμένης γραμμής.



Σχήμα 5.9

Στην προηγούμενη φάση όταν ο mapper γράφει τα αποτελέσματα του τα γράφει σε buffer της κύριας μνήμης και όταν τελειώσει την δουλειά μετά τα γράφει στο δίσκο όπως αναφέραμε σε πιο πάνω κεφάλαιο. Το μέγεθος αυτού του buffer είναι σταθερό και καθορίζεται στα configuration files με την λέξη κλειδί *mapreduce.task.io.sort.mb*

Όταν το buffer είναι σχεδόν γεμάτο (80%) , μπαίνει σε εφαρμογή το spilling phase παράλληλα χρησιμοποιώντας διαφορετικό thread. Σε περίπτωση που το buffer είναι αργό και φτάσει στο 100% full τότε το map job δεν μπορεί να εκτελεστεί και πρέπει να περιμένει.

Οι στάδια που ακολουθούνται:

1.Δημιουργεί ένα stream με το local filesystem.
2.Ταξινομεί τα chunks που βρίσκονται στο buffer
3.Το αποτέλεσμα της ταξινόμησης σπάζεται σε κομμάτια με βάση το key τα οποία θα σταλούν στους reducers
4.Τα κομμάτια γράφονται σειριακά σε ένα τοπικό file.

END MAP PHASE:
1.ταξινόμηση και spilling των επόμενων tuples
2. Κάνουμε shuffle ta δεδομένα στους reducers.

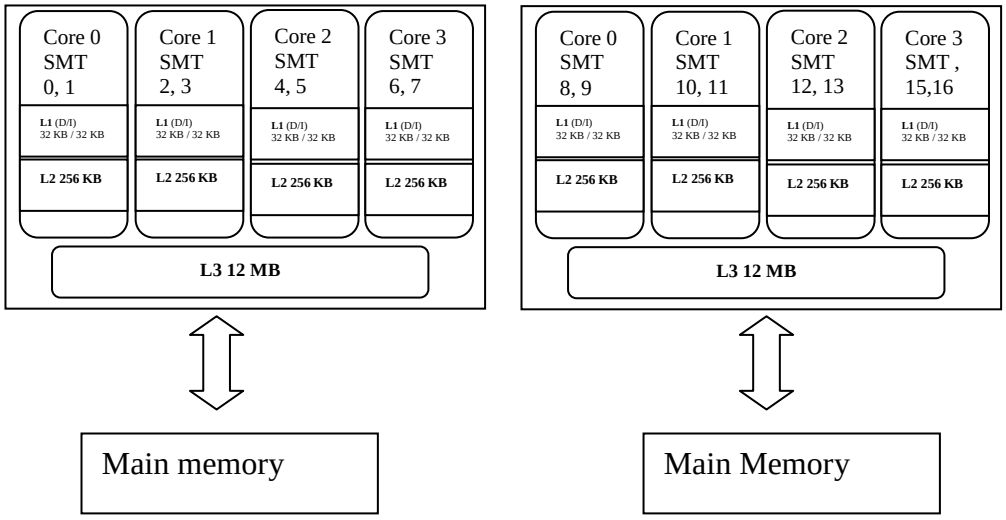
SHUFFLE:
Είναι η διαδικασία αποστολής των partitions στους κατάλληλους reducers.

5.8 Apache Mahout

Το Mahout προτάθηκε το 2008 ως subproject του Apache's Lucene project, που παρέχει την γνωστή ανοικτού πηγαίου κώδικα μηχανή αναζήτησης Lucene όπου το Mahout αποτελούσε την βιβλιοθήκη για τους αλγορίθμους mining, και τεχνικές ανάκτησης πληροφοριών που χρησιμοποιούσε η μηχανή αναζήτησης. Λίγο αργότερα η σημαντικότητα του Mahout το οδήγησε να αποτελέσει ένα άρτιο από μόνο του project.
Τώρα το Mahout είναι μια ανοικτού πηγαίου κώδικα βιβλιοθήκη η οποία υλοποιεί αλγόριθμους machine learning και συγκεκριμένα recommended engines, clustering και classification αλγόριθμους.
Σκοπός του Mahout ήταν να παρέχει scalability όσο αφορά τα δεδομένα , γι' αυτό το λόγο οι αλγόριθμοι που παρέχει υλοποιούνται πάνω σε κατανεμημένα συστήματα όπως το Hadoop framework.
Τέλος αξίζει να σημειωθεί ότι είναι γραμμένο σε Java και δεν παρέχει οποιοδήποτε user interfaceή prepackaged server , πάρα μόνο εργαλεία για να χρησιμοποιείται και να γίνεται adapt από τους χρήστες.

6.1 Περιβάλλον Εγκατάστασης

Για την εγκατάσταση των πειραμάτων χρησιμοποιήθηκε μια μηχανή του τμήματος ηλεκτρολόγων μηχανικών με 2 Numa Intel Xeon CPU E5620 με μέγιστη συχνότητα 2.40GHz ο κάθε ένας. Η μηχανή διαθέτει 32 GB κύριας μνήμης και σκληρό δίσκο 14TB. Το λειτουργικό σύστημα της μηχανής είναι Centos v6.0.1 64 bit. Το Hadoop που χρησιμοποιήθηκε είναι η 0.20.0 έκδοση που έτρεχε πάνω σε Java 6 (1.6) και Mahout έκδοση. Στην συγκεκριμένη διπλωματική λόγω της έλλειψης υποδομών εφαρμόσαμε pseudo-distributed setup όπου όλοι οι daemons τρέχουν πάνω στο ίδιο σύστημα.



Σχήμα 6.1

6.2 Εργαλεία που χρησιμοποιήθηκαν

Perf: Είναι Linux profiling εργαλείο το οποίο μετρά performance counters

Sar: Είναι System Monitor command που χρησιμοποιείται για να βρούμε διάφορα μετρικά του συστήματος όπως για παράδειγμα CPU activity, memory, paging και άλλα.

Free: Μέσω της free μπορούμε να πιάσουμε στατιστικά όσων αφορά την μνήμη για παράδειγμα πόση βρίσκεται σε χρήση, swap μνήμη και άλλα.

Top: Ένα από τα σημαντικότερα εργαλεία .Παρουσιάζει update πληροφορίες για τα cpu processes.

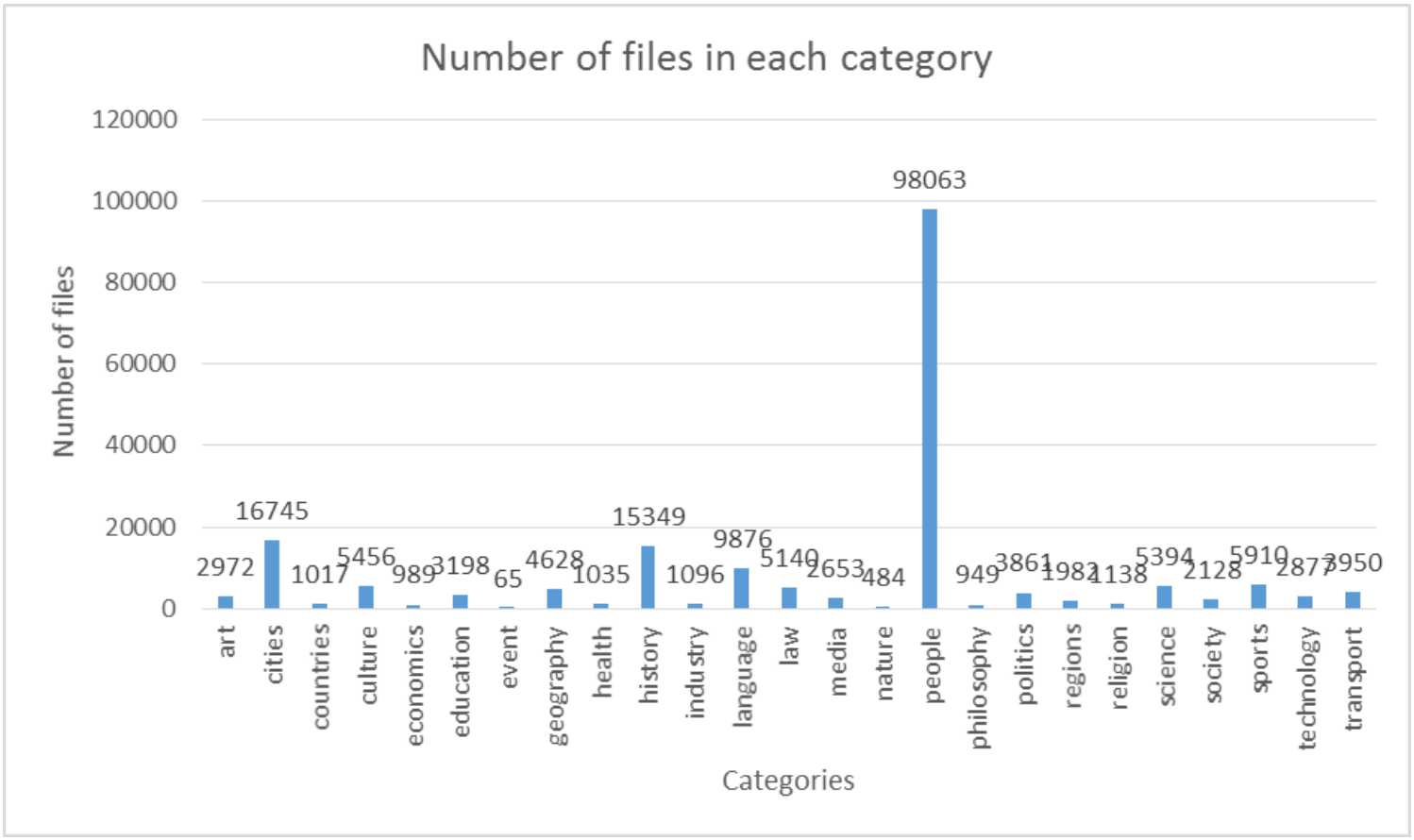
6.3 Δεδομένα που χρησιμοποιήθηκαν

Wikipedia (http://wikipedia.org):

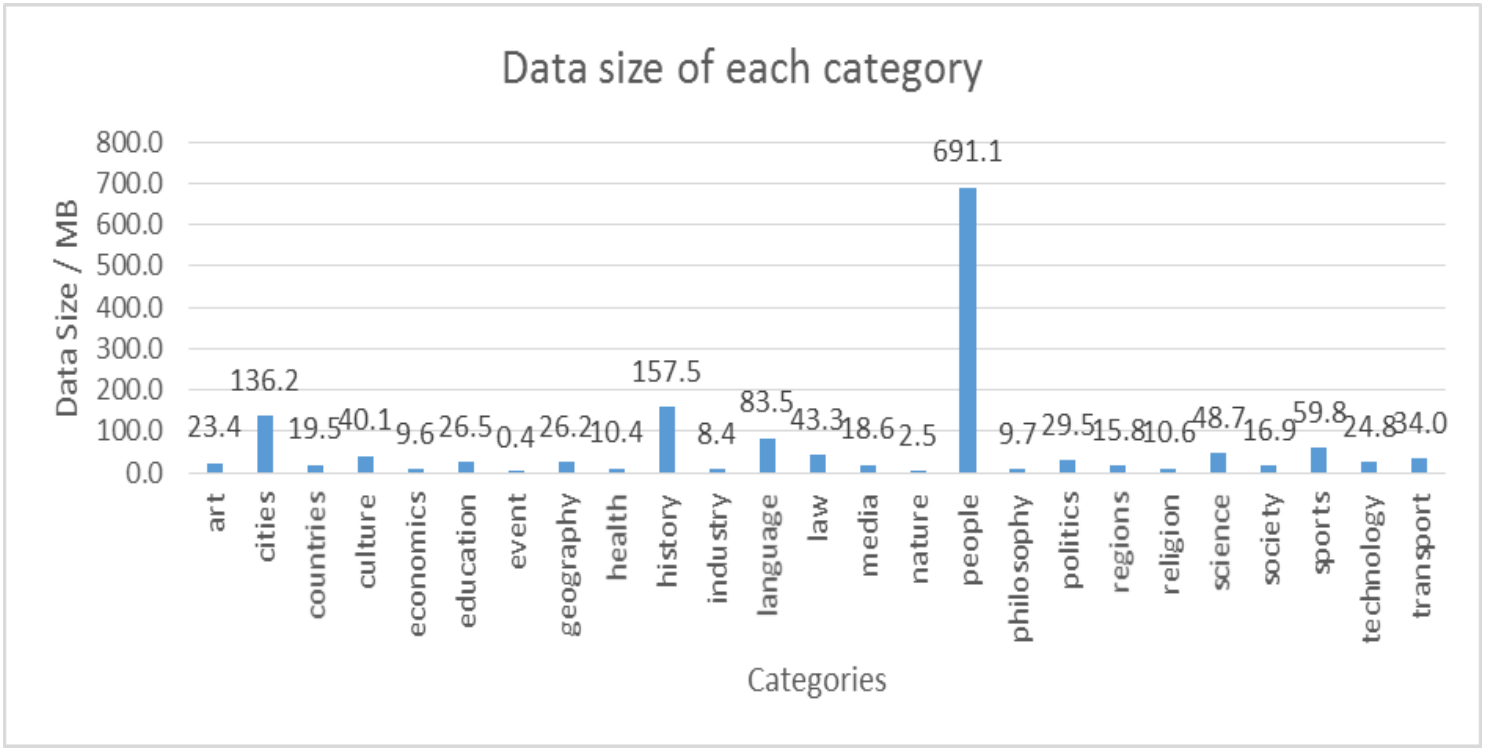
Είναι γνωστή online εγκυκλοπαίδεια που το περιεχόμενο της δημιουργείται και μορφοποιείται από τους χρήστες της. Σύμφωνα με κάποιες αναφορές το 2010 είχε πάνω από 3.2 εκατομμύρια άρθρα γραμμένα μόνο στα Αγγλικά που φτάνει σχεδόν τα 40GB σε χωρητικότητα, XML μορφή και είναι διαθέσιμα από το http://dumps.wikimedia.org/enwiki/ σύνδεσμο. Ο λόγος που διαλέξαμε αυτό το Data set είναι επειδή οργανώνεται και δημιουργείται από ανθρώπους πράγμα που μας δίνει ένα ρεαλισμό στα δεδομένα που χρησιμοποιούμε μιας και ακόμη δεν φτάσαμε στο σημείο να παράγουμε fake data sets που να έχουν τα χαρακτηριστικά των big data. Επίσης τα λάθη τα οποία περιέχουν τα δεδομένα είναι σημαντικά για την μελέτη μας.

Υπάρχουν διάφορα Data sets sizes διαθέσιμα από την Wikipedia. Για να μπορέσεις να αναλύσεις τα μεγάλα Data sets χρειάζεσαι μεγάλη υποδομή των εκατοντάδων μηχανών. Για τους δικούς μας σκοπούς χρησιμοποιήσαμε ένα μικρό υποσύνολο των δεδομένων αυτών. Συγκεκριμένα χρησιμοποιήσαμε ένα αρχείο των 15GB από το οποίο πήραμε μόνο όσο άρθρα ανήκαν σε μία από τις κατηγορίες και αφήσαμε μόνο το κείμενο των άρθρων. Αφαιρέσαμε δηλαδή τις μεταπληροφορίες του XML. Στις γραφικές φαίνονται αναλυτικά τα άρθρα που είχε η κάθε κατηγορία και το μέγεθος σε των συνολικών αρχείων κάθε κατηγορίας.

1. Art
2. Cities
3. Countries
4. Culture
5. Economics
6. Education
7. Event
8. Geography
9. Health
10. History
11. Industry
12. Language
13. Law
14. Media
15. Nature
16. People
17. Philosophy
18. Politics
19. Regions
20. Religion
21. Society
22. Sports
23. Technology
24. Transport



Γράφημα 6.1

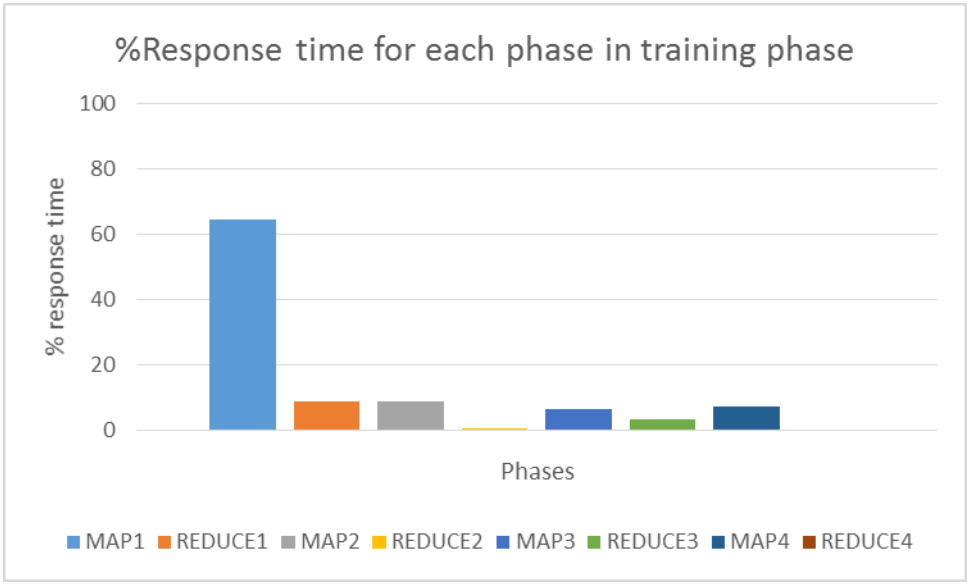


Γράφημα 6.2

6.4 Πειράματα και Συμπεράσματα

Πρώτα απ' όλα θέλαμε να καταλάβουμε πως δουλεύει ο αλγόριθμος και πια κομμάτια του καταναλώνουν ποιους πόρους. Λόγω του ότι το Hadoop default block είναι 64MB ακολούθησα τα ίδια βήματα αφήνοντας το block όπως ήταν και το αύξανα ανάλογα(128MB, 256 MB , 512 MB) .

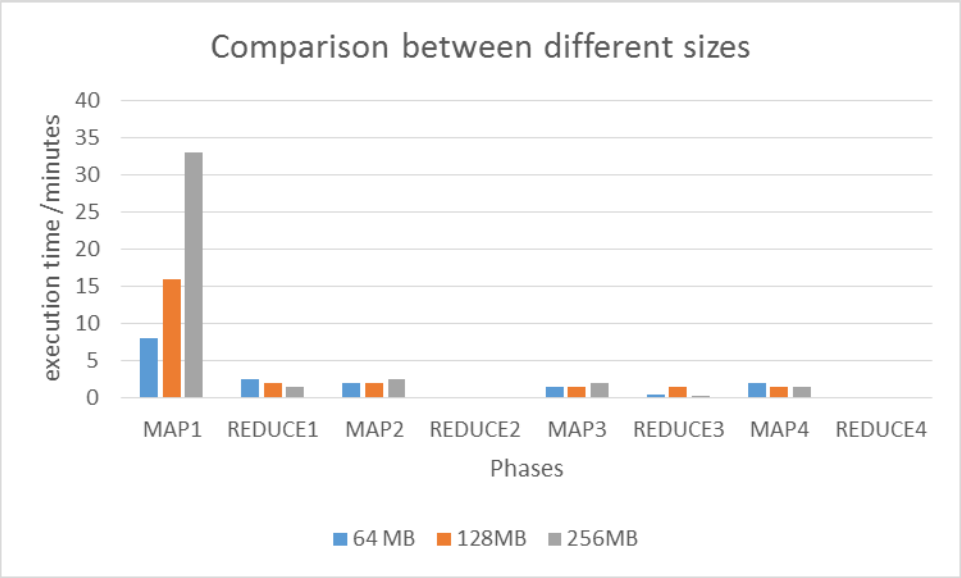
Για την εξαγωγή της πιο κάτω γραφικής παράστασης έτρεξα για τρία διαφορετικά μεγέθη block,64MB, 128MB και 256MB και πείρα τον μέσο όρο χρόνου και για τα τρία έτσι ώστε να έχω μια καλύτερη εικόνα για το τι γίνεται.



Γράφημα 6.3

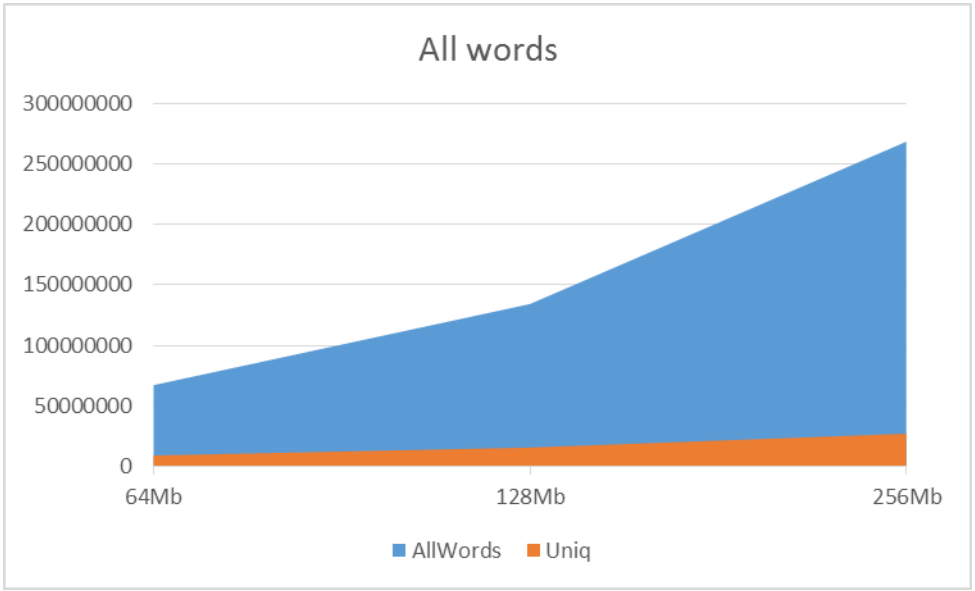
Ο αριθμός των mappers και reducers που τρέχουν στο σύστημα εξαρτούνται από το μέγεθος του αρχείου και από τον αριθμό που ορίσαμε εμείς στα configurations. Αυτό που γίνεται είναι πρώτα σπάζει το αρχείο σε κομμάτια μεγέθους block size .Στην συγκεκριμένη περίπτωση είναι 64MB , αν ο αριθμός των mappers που δώσαμε σαν παράμετρο εμείς είναι μικρότερος των αγνοεί αν είναι μεγαλύτερος τρέχει περισσότερους mappers.Επομένως εδώ η πρώτη φάση στα 64MB τρέχει με ένα core ,στα 128 με 2 και ούτω κάθε εξής.

Εδώ φαίνεται πως η πρώτη φάση του αλγόριθμου παίρνει το περισσότερο και έχει μεγάλη διαφορά σε σχέση με τα άλλα. Ο λόγος που το κάνει αυτό με βάση τον αλγόριθμο είναι εξαιτίας του ότι στην πρώτη φάση διαβάζει τα δεδομένα και υπολογίζει το TF-IDF μετρώντας πόσες φορές εμφανίζεται η κάθε λέξη και το κάθε feature. Αυτό όμως προκαλεί καθυστέρηση σε όλο το σύστημα γιατί αν και κατανεμημένο τα επόμενα βήματα (εκτός από τον επόμενο reducer με τον οποίο έχουν ένα είδος pipeline) δεν μπορούν να εκτελεστούν αν δεν τελειώσει ο mapper.



Γράφημα 6.4

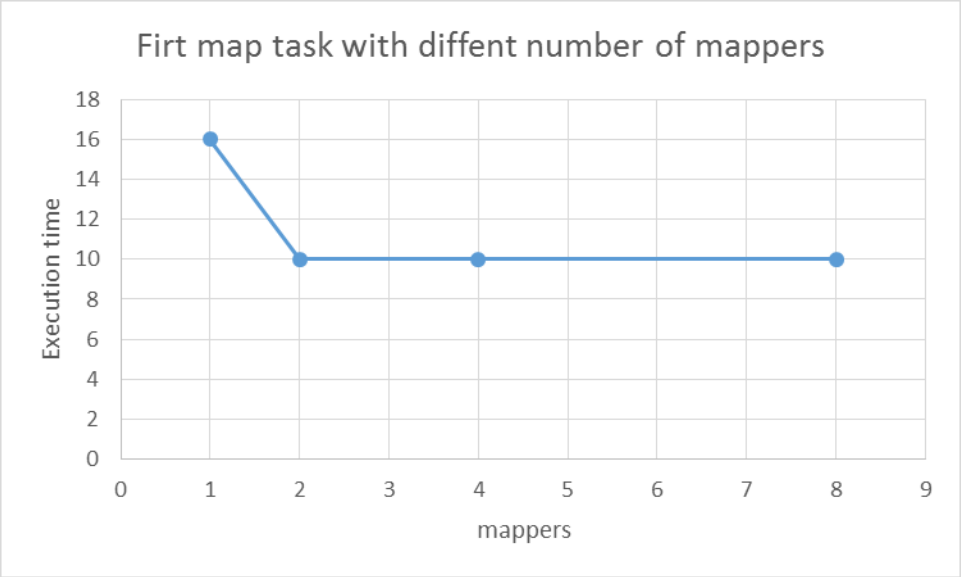
Αυτά τα αποτελέσματα μου δημιούργησαν κάποια ερωτήματα. Το πρώτο ερώτημα ήταν γιατί έχουμε τόσο μεγάλη διαφορά στη πρώτη φάση μεταξύ των διάφορων μεγεθών στα δεδομένα εισόδου και οι άλλες φάσεις είναι οι ίδιες. Αυτό που σκέφτηκα και πιστεύω πως συμβαίνει βρίσκεται στο τι κάνει η συγκεκριμένη εργασία. Αυτό που κάνει είναι να παίρνει ένα μεγάλο κείμενο 64MB και να βρίσκει πόσες φορές εμφανίζεται η κάθε λέξη μέσα στην κάθε κατηγορία και να το φιλάει. Άρα καταλαμβάνουμε πως ίσως το Dataset της εξόδου να είναι αρκετά μικρό και πανομοιότυπο σε όλα τα πιο πάνω πειράματα. Για να το ελέγξω αυτό είδα πόσες λέξεις στα 64MB είναι μοναδικές ,στα 128 το ίδιο και στα 256MB. Τα αποτελέσματα φαίνονται στην πιο κάτω γραφική παράσταση.



Γράφημα 6.5

Αυτό που καταλαβαίνουμε από την γραφική παράσταση είναι ότι αν και τα δεδομένα αυξάνονται οι μοναδικές λέξεις σε αυτά τα δεδομένα έχουν ελάχιστη αύξηση κάθε φορά με αποτέλεσμα να επηρεάζεται από την ποσότητα των δεδομένων μας μόνο η πρώτη φάση. Όπως έχουμε πει και πιο πάνω όταν γίνεται το word frequency στην πρώτη φάση λέξεις με πανομοιότυπο prefix θεωρούνται σαν μια. Εγώ αυτό δεν το έκανε στην πιο πάνω προσέγγιση. Δηλαδή η λήξη Μάτι και Μάτια θα θεωρούνταν δύο διαφορετικές λέξεις. Επομένως στην πραγματικότητα οι λέξεις είναι ακόμα πιο λίγες.

Το δεύτερο μου ερώτημα ήταν τι θα γινόταν αν έτρεχα την πρώτη φάση , το πρώτο map job σε περισσότερους από ένα πυρήνες. Έτρεξα το 128MB με 1 ,2,4 και 8 mappers και παρατήρησα πως από τον 1 στους 2 mappers είδα διαφορά αλλά μετά καμία.

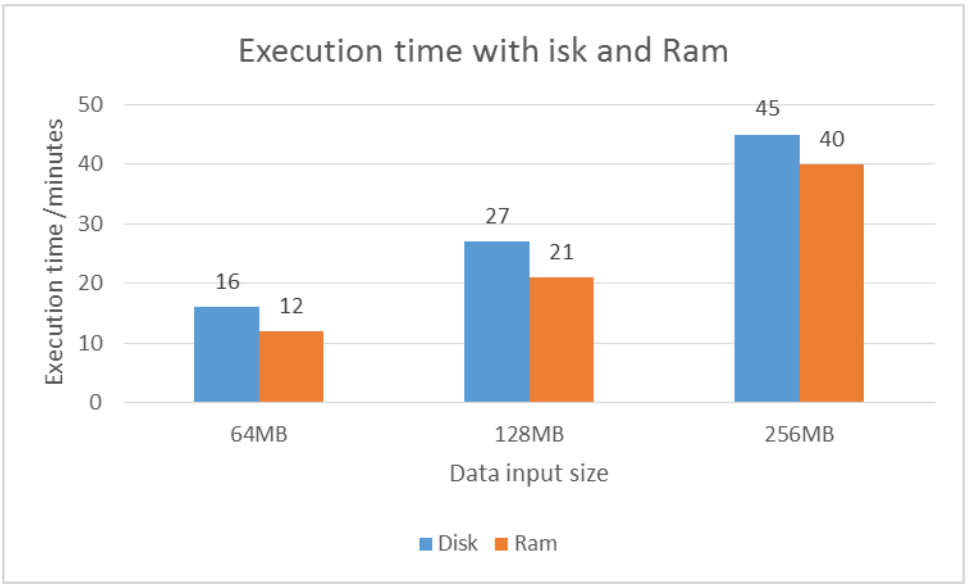


Γράφημα 6.6

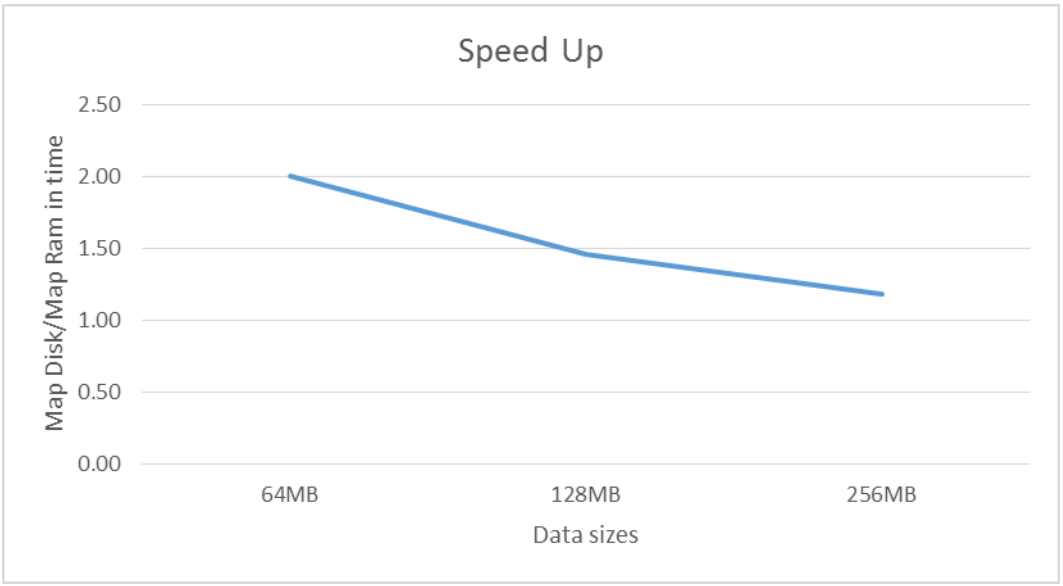
Στην πιο κάτω γραφική παράσταση φαίνεται ο χρόνος που χρειαζόταν για να τρέξουν και τα τέσσερα στάδια. Βλέπουμε πως η αύξηση είναι γραμμική δίνοντας μας το scalability του συστήματος. Φαίνεται και για ακόμα μια φορά εδώ πως η μοναδική διαφορά στο χρόνο τους είναι αυτή του πρώτου map job.

Γράφημα 6.7

Στην συνέχεια ήθελα να δω τι θα συμβεί αν αντί δίσκο κάναμε τα πάντα μέσω της κύρια μνήμης. Ίσως μια προσομοίωση των in –memory frameworks. Παρατηρούμε πως αν και η διαφορά των χρόνων όταν χρησιμοποιούμε RAM και δίσκο συνήθως είναι μεγάλη εδώ φαίνεται πως η διαφορά δεν είναι και τόσο μεγάλη. Ο λόγος φαίνεται να έγκειται στο γεγονός ότι μόνο η πρώτη φάση και η τελευταία κάνει κάτι διαφορετικό όταν τα δεδομένα βρίσκονται στον δίσκο ή στη μνήμη. Η τελευταία φάση όπως είδαμε και πριν παίρνει μηδαμινό χρόνο. Επομένως μπορούμε να θεωρήσουμε ότι η διαφορά των δύο χρόνων είναι μείωση στην πρώτη φάση η αναλογία φαίνεται στην πιο κάτω γραφική παράσταση.



Γράφημα 6.8

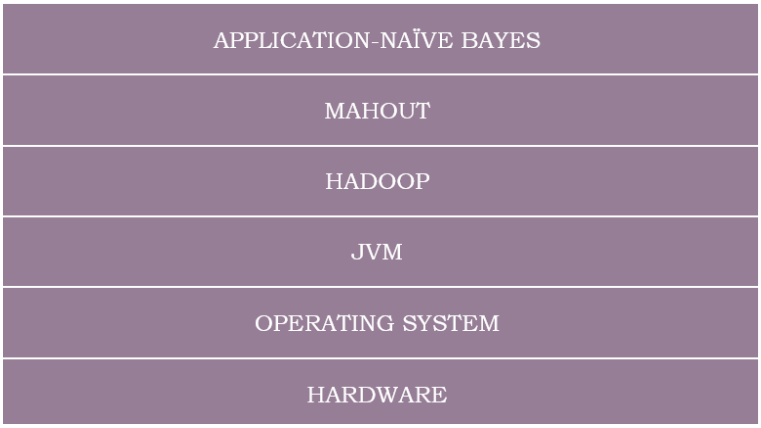

Speed Up
Map Disk/Map Ram in time
2.50
2.00
1.50
1.00
0.50
0.00
64MB
128MB
256MB
Data sizes

Γράφημα 6.9

Κεφάλαιο 7 Μελλοντική Εργασία

7.1 Μελλοντική Εργασία

1. Το συγκεκριμένο σύστημα που προσπαθούμε να αναλύσουμε επηρεάζεται από πολλούς τομείς. Όπως φαίνεται και στο σχεδιάγραμμα μια πραγματική κατανόηση του τι γίνεται απαιτεί εξέταση πολλών στρωμάτων. Ίσως μια πιο βαθιά κατανόηση του συστήματος θα μπορούσε να δώσει ακόμα περισσότερες πληροφορίες στο θέμα των Data Analytics εφαρμογών.



2. Μια επιπλέον μελλοντική εργασία θα ήταν να φεύγαμε από τα όρια του pseudo-distributed συστήματος το οποίο χρησιμοποιούμαι και να εξετάζαμε τι γίνεται σε πραγματικό cluster όπου μπαίνουν στο παιχνίδι και παράγοντες όπως network overhead

3. Τέλος θα μπορούσαμε να χρησιμοποιήσουμε άλλα frameworks ή άλλες εφαρμογές Data Analytics και να δούμε πιες είναι οι διαφορές τους , που είναι βέλτιστο το ένα και που το άλλο.

Βιβλιογραφία

[1] Understanding Big Data: Analytics for enterprise class Hadoop and streaming data

[2] How big is a cloud, [http://www.extremetech.com/computing/129183-how-big-is-the-cloud]

[3] Veracity, [http://www.paxata.com/missing-v-data-veracity]

[4] Ventana research perspective efficient data preparation paxata 2014, [http://www.paxatadev.com/sites/default/files/ventana-research-perspective-efficient-data-preparation-paxata-2014.pdf]

[5] Map Reduce ,[http://research.google.com/archive/mapreduce.html]

[6] GFS,[http://research.google.com/archive/gfs.html]

[7] Michel Ferdman, Bsbak Falsafi, A study of Emerging Scale out Workloads on Modern Hardware

[8] Instagram ,[https://instagram.com/press/]

[9] AnatomyMapReduce [http://ercoppa.github.io/HadoopInternals/AnatomyMapReduceJob.html]

[10] Rui Han and , Xiaoyi Lu : On Big Data Benchmarking Department of Computing, Imperial College London ,Ohio State University (2014)

[11] Yanpei Chen, UC Berkeley: We Don't Know Enough to make a Big Data Benchmark Suite - An Academia-Industry View

[12]Big Data Bench, A Big Data benchmark suite, ICT, Chinese Academy of Sciences, [http://prof.ict.ac.cn/BigDataBench]

[13] The HiBench Benchmark Suite: Characterization of the MapReduce-Based Data Analysis Shengsheng Huang, Jie Huang, Jinquan Dai, Tao Xie, and Bo Huang Intel China Software Center, Shanghai, P.R. China, 200241

[14] Narayanan, R. ; Electr. Eng. & Comput. Sci., Northwestern Univ., Evanston, IL ; Ozisikyilmaz, B. ; Zambreno, J. ; Memik, G. MineBench: A Benchmark Suite for Data Mining Workloads

[15] Andrea De Mauro, Marco Greco and Michele Grimaldi
What is big data? A consensual definition and a review of key research topics

[16] http://www01.ibm.com/support/knowledgecenter/linuxonibm/liacf/oprofbasicoperf.htm

[17] http://www.slideshare.net/ZxMYS/hi-tune-sharing