ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. ΔΙΚΤΥΟΚΕΝΤΡΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

“NoSQL databases : Ποιοτική και Ποσοτική Σύγκριση μεταξύ των Cassandra, BaseX και Mongodb”

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παναγοπούλου Μαριάνθη
(ΜΕ11065)

Επιβλέπων Καθηγητής : Μαρίνος Θεμιστοκλέους

Αθήνα, Ιούλιος 2015

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά όλους εκείνους που συνέβαλαν με οποιονδήποτε τρόπο στην εκπόνηση της διπλωματικής μου εργασίας και ιδιαίτερες ευχαριστίες εκφράζω :

Στον επιβλέποντα καθηγητή μου κ. Μαρίνο Θεμιστοκλέους για την ανάθεση της εργασίας και την επίβλεψη του

Στον συνεπιβλέποντα καθηγητή μου κ. Δημοσθένη Κυριαζή για την πολύτιμη και αδιάκοπη βοήθεια του, τις συμβουλές του και την καθοδήγησή του όλους αυτούς τους μήνες που διήρκησε η συγγραφή αυτής της εργασίας

Περίληψη

Με το πέρασμα των χρόνων έχει παρατηρηθεί μια ραγδαία και συνεχής εξέλιξη των υπολογιστικών συστημάτων καθώς και των εφαρμογών που υπάρχουν πλέον στην αγορά, καθώς και μια εκθετική αύξηση του όγκου των δεδομένων που δημιουργείται από χρήστες, διαφόρων ειδών συστήματα και αισθητήρες. Στα πλαίσια αυτά τα σχεσιακά μοντέλα των βάσεων δεδομένων που επικρατούσαν μέχρι τώρα (SQL), χάνουν έδαφος λόγω της αδυναμίας τους να ανταποκριθούν στις σύγχρονες απαιτήσεις. Στην θέση αυτών έρχονται να μπουν μιας νέας γενιάς βάσεις δεδομένων που ονομάζονται NoSQL. Οι βάσεις αυτές παρουσιάζουν αρκετά πλεονεκτήματα όσον αφορά τις νεοαποκτηθείσες τάσεις της αγοράς, αλλά ταυυτόχρονα και μειονεκτήματα σε σχέση με τον προκάτοχό τους. Η παρούσα εργασία ασχολείται με τη νέα αυτή γενιά βάσεων δεδομένων και συγκεκριμένα έχουν επιλεχθεί τρεις βάσεις για να γίνει μια ποιοτική σύγκριση στα χαρακτηριστικά τους, στο τι προσφέρουν, αλλά και στο σε τι υστερούν σε σχέση με μια σχεσιακή βάση. Τέλος επιλέχθηκε ένα benchmarking tool,ώστε να γίνουν κάποια τεστ με σκοπό να αξιολογηθεί η απόδοσή τους με πραγματικά δεδομένα.

Περιεχόμενα

Κατάλογος Εικόνων/Σχημάτων

Κατάλογος Πινάκων

1. Εισαγωγή

1.1. Σκοπός και Αντικείμενο Εργασίας

Σκοπός της διπλωματικής εργασίας αυτής είναι να κάνει μια εισαγωγή στις NoSQL βάσεις δεδομένων. Θα γίνει περιγραφή της αρχιτεκτονικής τους και ανάλυση των σημείων στα οποία υπερτερούν σε σχέση με τα παραδοσιακά σχεσιακά μοντέλα βάσεων δεδομένων. Στη συνέχεια θα επιλεχθούν και θα αναλυθούν περαιτέρω 2 βάσεις της οικογένειας των NoSQL οι οποίες ανήκουν σε διαφορετικές επιμέρους κατηγορίες. Θα γίνει αναφορά στην αρχιτεκτονική τους και στο database model στο οποίο είναι βασιμένες, θα περιγραφούν οι διάφορες δυνατότητες που παρέχουν και θα αναπτυχθούν τα πλεονεκτήματα και τα μειονεκτήματά τους. Τέλος θα γίνει μια συγκριτική μελέτη μεταξύ των 2 βάσεων αυτών. Στο τελευταίο κομμάτι της εργασίας, θα εγκαταστηθούν 2 NoSQL βάσεις στο ίδιο σύστημα, και θα γίνει μελέτη απόδοσης με διάφορα τεστς τα οποία βασίζεται σε κάποιο benchmarking tool.

1.2.Ποιο είναι το πρόβλημα και γιατί στραφήκαμε σε NoSQL

Τα τελευταία χρόνια έχει παρατηρηθεί ότι η χρήση των σχεσιακών βάσεων δεδομένων (relational DBs), σε πολλές περιπτώσεις, οδηγεί σε προβλήματα στην μοντελοποίηση των δεδομένων και εισάγει περιορισμούς στο horizontal scalability σε πολλούς εξυπηρετητές (servers) και σε μεγάλες ποσότητες δεδομένων. Υπάρχουν δύο τάσεις που έχουν επιστήσει την προσοχή της διεθνούς κοινότητας λογισμικού:

1. Η εκθετική αύξηση του όγκου των δεδομένων που δημιουργείται από χρήστες, διαφόρων ειδών συστήματα και αισθητήρες, επιταχύνθηκε περαιτέρω από τη συγκέντρωση του μεγάλου μέρους των δεδομένων αυτών σε μεγάλα κατανεμημένα συστήματα όπως της Amazon, της Google και άλλων υπηρεσιών cloud.
2. Η αυξανόμενη αλληλεξάρτηση και η πολυπλοκότητα των δεδομένων επιταχύνονται από το Διαδίκτυο, το Web2.0, τα κοινωνικά δίκτυα και από την ανοικτή και τυποποιημένη πρόσβαση σε πηγές δεδομένων από ένα μεγάλο αριθμό διαφορετικών συστημάτων.

Οι υπολογιστικές και αποθηκευτικές απαιτήσεις των εφαρμογών, όπως τα Big Data Analytics [1], Business Intelligence [2] και social networking, για μεγέθη δεδομένων τύπου peta-byte, ώθησαν τις βάσεις δεδομένων τύπου SQL στα όριά τους [3]. Αυτό οδήγησε στην ανάπτυξη βάσεων που είναι οριζόντια επεκτάσιμες, μη σχεσιακού τύπου, που ονομάζονται No-SQL, όπως το Bigtable της Google [4] και οι ανοιχτού κώδικα εφαρμογές του HBase [5]

και Cassandra (Facebook) [6]. Η εμφάνιση των κατανεμημένων βάσεων, τύπου key-value, όπως οι Cassandra και Voldemort [7], αποδεικνύει την αποτελεσματικότητα και την αποδοτικότητα στο κόστος των προσεγγίσεών τους [8]. Οι βασικοί περιορισμοί με τις RDBMS είναι ότι είναι δύσκολο το «ταίριασμά» τους με εφαρμογές Cloud, Data warehousing, Grid και Web 2.0 [9].

Το αυστηρά σχεσιακό σχήμα συνιστά συνήθως εμπόδιο για δικτυακές εφαρμογές, όπως τα blogs, τα οποία αποτελούνται από πολλά διαφορετικά είδη attributes. Κείμενο, σχόλια, φωτογραφίες, βίντεο, πηγαίος κώδικας και άλλες πληροφορίες πρέπει να αποθηκευτούν μέσα σε πολλαπλούς πίνακες (tables). Δεδομένου ότι αυτές οι εφαρμογές web είναι πολύ ευέλικτες, οι υποκείμενες βάσεις δεδομένων πρέπει να είναι αντίστοιχα ευέλικτες, ώστε να μπορούν να τις υποστηρίξουν [10]. Τα NoSQL συστήματα εμφανίζουν την ικανότητα να αποθηκεύουν και να κατατάσσουν (Indexing) μεγάλα σύνολα δεδομένων, ενώ την ιδια στιγμήεπιτρέπουν ταυτόχρονα αιτημάτα των χρηστών [11].

Οι σύγχρονες web εφαρμογές έχουν εξελιχθεί σε μεγάλο βαθμό σε σχέση με τα προηγούμενα χρόνια. Τα δεδομένα που διαχειρίζονται έχουν μεγαλύτερο όγκο και είναι πιο πολύπλοκα. Αυτό έχει σαν συνέπεια οι αλγόριθμοι που απαιτούνται για την διαχείρησή τους, καθώς και τα ερωτήματα να γίνονται πολυπλοκότερα και ο χρόνος επεξεργασίας και απόκρισης του συστήματος να αυξάνεται δραματικά. Τα χαρακτηριστικά αυτά οδήγησαν σε νέες απαιτήσεις:

(1) Απαίτηση για **κλιμάκωση**: να υπάρχει η δυνατότητα προσθήκης νέων servers όταν οι εφαρμογές σχετίζονται με μεγάλους όγκους δεδομένων, καθώς και να υπάρχει υψηλός ρυθμός διεκπεραίωσης επερωτήσεων (τόσο σε reads όσο και σε updates).

(2) Απαίτηση για **διαθεσιμότητα**: Η εφαρμογή πρέπει να είναι πάντα διαθέσιμη, ακόμη και σε απρόβλεπτες συνθήκες φόρτου, και όλα τα δεδομένα να είναι πάντα διαθέσιμα (ακόμα και αν μερικοι servers τεθούν εκτός λειτουργίας).

(3) Απαίτηση για **απόδοση**: οι καθυστερήσεις να είναι όσο πιο χαμηλές γίνεται, ακόμα και σε συμβατικό hardware [12].

1.3.Τι υπάρχει ήδη και που υστερεί - αδυναμία σχεσιακού μοντέλου

Το σχεσιακό μοντέλο αποτελεί ακόμη και σήμερα το πιο διαδεδομένο μοντέλο που υποστηρίζει τις περισσότερες εφαρμογές διαχειριστικού τύπου (τραπεζικά συστήματα, συστήματα κράτησης θέσεων, κλπ). Υπάρχει μια ευρεία αποδοχή του μοντέλου αυτού στην

αγορά και στον επιχειρηματικό κόσμο και χρησιμοποιούνται για εφαρμογές όπως η μισθοδοσία, η επεξεργασία παραγγελιών, συστήματα κρατήσεων κλπ. Η επικράτηση τους αυτή στην αγορά οφείλεται στα εξής χαρακτηριστικά που προσφέρουν :

Υποστηρίζουν μια απλή δομή δεδομένων (πίνακες), περιορίζουν την επανάληψη (duplication) των δεδομένων, αποφεύγουν τις ασυνέπειες των στοιχείων που αποθηκεύονται, παρέχουν φυσική και λογική ανεξαρτησία των δεδομένων σε μεγάλο βαθμό, υποστηρίζουν adhoc ερωτήματα με την χρήση της γλώσσας SQL [13], βελτιστοποίηση και προσαρμογή επερωτήσεων (Joins), μόνιμη αποθήκευση, υποστηρίζουν συναλλαγές και τις ιδιότητες ACID (Ατομικότητα, Συνοχή, Απομόνωση, Μονιμότητα/Ανθεκτικότητα), Ασφάλεια/Πιστοποίηση.

Ωστόσο, υπάρχουν σύγχρονες εφαρμογές που θέτουν μεγάλες απαιτήσεις με αποτέλεσμα η υποστήριξη των εφαρμογών αυτών από το σχεσιακό μοντέλο να είναι αρκετά δύσκολη. Το σχεσιακό μοντέλο εμφανίζει αρκετά μειονεκτήματα και περιορισμούς στην εσωτερική δομή της βάσης, αλλά και στους μηχανισμούς με τους οποίους οι επιχειρηματικές εφαρμογές αποκτούν πρόσβαση στα δεδομένα. Μεγάλο μειονέκτημα που παρουσιάζουν οι σχεσιακές βάσεις είναι ότι έχουν κακή κλιμάκωση.Ένα ακόμα μειονέκτημα των σχεσιακών βάσεων δεδομένων είναι το ακριβό για τη δημιουργία και τη διατήρηση του συστήματος της βάσης δεδομένων. Κατά τη σχεδίαση της βάσης δεδομένων , θα πρέπει να καθορίζετε εξαρχής η ποσότητα και το schema των δεδομένων που μπορούν να χωρέσουν σε ένα πεδίο.

Αδυναμία ή φτωχή αναπαράσταση του πραγματικού κόσμου : Η διαδικασία της κανονικοποίησης οδηγεί στη δημιουργία σχέσεων που δεν αντιστοιχούν σε οντότητες του πραγματικού κόσμου, κάτι το οποίο έχει σαν επακόλουθο και την δυσκολία στην αναπαράσταση συνθέτου τύπου δεδομένων. Η κατατεμάχιση των δεδομένων σε πολλούς πίνακες οδηγεί στην εκτέλεση πολλών χρονοβόρων πράξεων σύνδεσης. Ομοιογένεια. Το σχεσιακό μοντέλο ορίζει ότι κάθε γραμμή ενός πίνακα πρέπει να αποτελείται από τις ίδιες στήλες, ενώ κάθε στήλη του πίνακα πρέπει να δέχεται τιμές από το ίδιο πεδίο ορισμού. Οι δύο αυτές ιδιότητες καλούνται οριζόντια και κάθετη ομοιογένεια. Η δομή αυτή του πίνακα είναι αρκετά περιοριστική για αντικείμενα του πραγματικού κόσμου τα οποία έχουν πολύπλοκη δομή. Οι αλλαγές στο σχήμα της ΒΔ είναι χρονοβόρες. Αν απαιτηθεί αλλαγή στο σχήμα της ΒΔ θα πρέπει ο διαχειριστής να διακόψει προσωρινά τη λειτουργία του συστήματος και επιπλέον τα προγράμματα εφαρμογής πρέπει να διαμορφωθούν αναλόγως. Περιορισμένες λειτουργίες: Τα σχεσιακά ΣΔΒΔ διαθέτουν ένα περιορισμένο σύνολο λειτουργιών επί των δεδομένων, το οποίο προσδιορίζεται από την SQL. Σήμερα

απαιτείται η δυνατότητα ορισμού νέων τύπων δεδομένων και λειτουργιών. Τα προαναφερθέντα προβλήματα οδήγησαν στη μελέτη νέων μεθόδων μοντελοποίησης και διαχείρισης ώστε να είναι εφικτή η αποτελεσματική και αποδοτική διαχείριση των σύγχρονων δεδομένων.

Στα μειονεκτήματα των σχεσιακών βάσεων, βασίζεται (ή ίσως καλύτερα έχει εμπνευστεί)η σχεδιαστική φιλοσοφία των NoSQL συστημάτων. Τα NoSQL είναι κατανεμημένα συστήματα που είναι κλιμακώσιμα, ανθεκτικά, προσαρμοστικά, εύκολα διαχειρίσιμα, αποδοτικά, με υψηλή διαθεσιμότητα και επιτρέπουν παράλληλη επεξεργασία [12]. Οι μεγαλύτεροι δικτυακοί τόποι όπως οι Google, Amazon, Facebook, twitter, digg, reddit, Linkedin, Sourceforge, bing κλπ, δεν χρησιμοποιούν πια σχεσιακές βάσεις και έχουν στρέψει την προσοχή τους στην τεχνολογία των NoSQL συστημάτων.

2. Βιβλιογραφική Ανασκόπηση

2.1. Θεώρημα CAP και ACID ιδιότητες

Προκειμένου να διασφαλιστεί η ακεραιότητα των δεδομένων, τα περισσότερα από τα κλασικά συστήματα βάσεων δεδομένων βασίζονται σε συναλλαγές. Αυτό εξασφαλίζει τη συνοχή των δεδομένων σε όλες τις περιπτώσεις της διαχείρισης τους. Το σύνολο των ιδιοτήτων το οποίο εγγυάται ότι οι συναλλαγές στη βάση δεδομένουν λειτουργούν αξιόπιστα είναι γνωστό με το ακρωνύμιο ACID (Atomicity, Consistency, Isolation, Durability - ατομικότητα, συνεκτικότητα, απομόνωση, διάρκεια ζωής) [14].

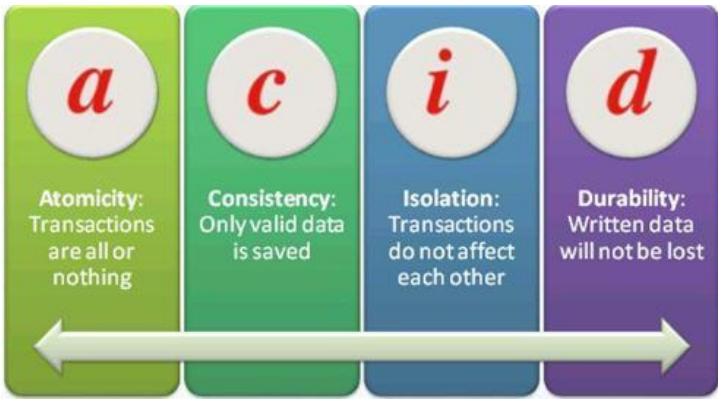
Ατομικότητα (Atomicity): Η Ατομικότητα απαιτεί η τροποποίηση που θα γίνει στην βάση να τηρεί τον κανόνα όλα ή τίποτα. Κάθε συναλλαγή η οποία είναι ατομική ονομάζεται έτσι επειδή αν ένα μέρος της αποτύχει, αποτυγχάνει όλη η συναλλαγή και η βάση μένει όπως ήταν πριν εκτελεστεί η συναλλαγή [15].

Συνέπεια (Consistency): Η ιδιότητα της Συνέπειας διασφαλίζει ότι η βάση διατηρείται σε μια συνεπή κατάσταση, συγκεκριμένα λέει ότι κάθε συναλλαγή θα οδηγεί την βάση δεδομένων από την μια συνεπή κατάσταση στη άλλη [15].

Απομόνωση (Isolation): Η Απομόνωση αναφέρεται στην απαίτηση ότι όλες οι ενέργειες δεν μπορούν να έχουν πρόσβαση ή να δουν δεδομένα τα οποία τροποποιούνται εκείνη την στιγμή από μια συναλλαγή η οποία δεν έχει ακόμα ολοκληρωθεί. Κάθε συναλλαγή δεν πρέπει να ξέρει αν υπάρχουν άλλες συναλλαγές που εκτελούνται ταυτόχρονα, αλλά να

περιμένουν την ολοκλήρωση μιας συναλλαγής ώστε να δουν/τροποποιήσουν τα δεδομένα τα οποία χρειάζεται και η άλλη συναλλαγή [15].

Μονιμότητα (Durability): Η Μονιμότητα εγγυάται στον χρήστη ότι αν τελειώσει μια συναλλαγή επιτυχώς τότε τα αποτελέσματα της δεν θα χαθούν. Οι αλλαγές που έχει κάνει η συναλλαγή δεν θα χαθούν έστω και αν κρασάρει το σύστημα και ότι όλες οι προϋποθέσεις ακεραιότητας ισχύουν, έτσι ώστε το σύστημα δεν θα χρειαστεί να ακυρώσει τη συναλλαγή αυτή [15].



Εικόνα 1:ACID ιδιότητες [https://www.bizcloudnetwork.com/]

Ωστόσο, η απομάκρυνση από τα συστήματα που έχουν ACID ιδιότητες, έχει αποδειχθεί ότι δημιουργεί διαφόρων ειδών προβλήματα. Συγκρούσεις προκύπτουν μεταξύ των διαφόρων πτυχών της υψηλής διαθεσιμότητας σε κατανεμημένα συστήματα, οι οποίες δεν είναι πλήρως επιλύσιμες. Την κατάσταση αυτή περιγράφει το θεώρημα CAP [16]:

Το 2000, ο Eric A. Brewer υποστήριξε ότι κάθε διαμοιρασμένο σύστημα μπορεί να έχει μόνο δύο από τις εξής τρεις βασικές ιδιότητες:

• **Strong Consistency –Ισχυρή Συνέπεια-Συνοχή**: Όλοι οι χρήστες να βλέπουν την ίδια έκδοση των δεδομένων, ακόμα και κατά τη διάρκεια ενημερώσεων της βάσης

• **High Availability – Υψηλή Διαθεσιμότητα**: Όλοι οι χρήστες να μπορούν να βρουν πάντα τουλάχιστον ένα αντίγραφο των ζητούμενων δεδομένων, ακόμη και αν ορισμένα από τα μηχανήματα σε ένα cluster έχουν πέσει.

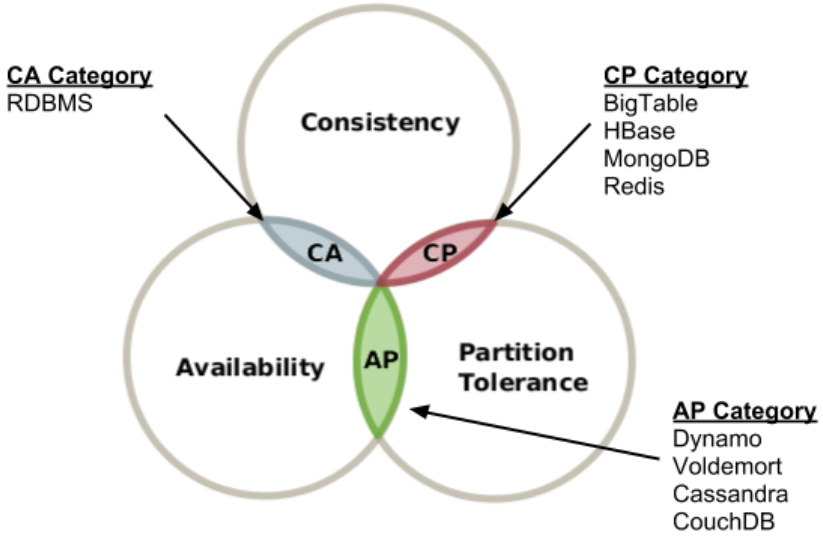
• **Partition Tolerance - Ανοχή στις κατατμήσεις/διαμέριση**: το σύστημα πρέπει να συνεχίσει να λειτουργεί ακόμα και αν έχει χαθεί οποιοσδήποτε αριθμός μηνυμάτων που αποστέλλονται μεταξύ των κόμβων.

Το θεώρημα CAP υποθέτει ότι μόνο δύο από τις τρεις διαφορετικές ιδιότητες της κλιμάκωσης ενός συστήματος είναι δυνατόν να επιτευχθούν πλήρως ταυτόχρονα. Αυτό βέβαια δεν σημαίνει ότι πχ η επιλογή της διαθεσιμότητας θα έχει σαν αποτέλεσμα να μην εξασφαλίζεται καμία συνοχή στο σύστημα. Με βάση λοιπόν το θεώρημα αυτό, δημιουργούνται 3 συνδυασμοί :

CA: Στην ομάδα αυτή υπάγονται μικρά clusters με μικρό partitioning στα δεδομένα, ένα ταυτόχρονα εγγυάται η μεγάλη διαθεσιμότητα και η συνέπεια.

CP: Τα δεδομένα εδώ δεν είναι συνεχώς προσβάσιμα, αλλά υπάρχει συνέπεια και αντοχή στην κλιμάκωση.

AP: Τα δεδομένα είναι συνεχώς διαθέσιμα με τον κίνδυνο να μην είναι πάντα ενημερωμένα.



Εικόνα 2: Χαρακτηριστικά του θεωρήματος CAP [https://disqus.com/home/discussion/eincs/cap_theorem_eincs_84/]

Πολλές από τις NoSQL βάσεις δεδομένων έχουν χαλαρώσει τις απαιτήσεις σχετικά με τη συνοχή, προκειμένου να επιτύχουν καλύτερη διαθεσιμότητα και διαμέριση. Αυτό είχε ως αποτέλεσμα συστήματα που ονομάζονται BASE (Basically Available, Soft-state, Eventually consistent - Βασικά Διαθέσιμα, soft-state, τελικά συνεπή). Αυτά δεν υποστηρίζουν συναλλαγές με την κλασσική έννοια και εισάγουν περιορισμούς στο μοντέλο δεδομένων για να καταστεί καλύτερο το σχήμα κατάτμησης. Οι Han et al. (2011) κατηγοριοποιούν τις NoSQL βάσεις δεδομένων σύμφωνα με το θεώρημα της CAP [17].

2.2. NoSQL βάσεις δεδομένων

2.2.1. Τι είναι τα NoSQL συστήματα

Ο όρος NoSQL, μεταφράζεται ως «Not Only SQL» και πρωτοεμφανίστηκε στις αρχές του 2009 σε μια συνάντηση στο SanFrancisco, κάτα την οποία παρουσιάστηκαν κάποιες «νέες» βάσεις δεδομένων [18]. Σαφής ορισμός σχετικά με ένα σύστημα NoSQL δεν υπάρχει. Αναφέρεται σε μια ευρεία ομάδα συστημάτων διαχείρισης βάσεων δεδομένων (database management system) που το κύριο χαρακτηριστικό τους είναι το ότι δεν τηρούν το RDBMS μοντέλο (Relational Database Management System), το οποίο και χρησιμοποιείται στην συντριπτική πλειοψηφία των περιπτώσεων, καθώς επίσης χρησιμοποιούν διαφορετικό από τον κλασικό τρόπο (SQL) για την διαχείριση και επεξεργασία των δεδομένων μέσα στη βάση (data manipulation) [19].

Τα NoSQL συστήματα είναι κατανεμημένες, μη σχεσιακές (non-relational) βάσεις, σχεδιασμένες για αποθήκευση δεδομένων μεγάλης κλίμακας και παράλληλη επεξεργασία δεδομένων μοιρασμένα σε έναν μεγάλο αριθμό από servers. Οι NoSQL βάσεις δεδομένων γενικώς δεν χρησιμοποιούν κάποιο δομημένο σύστημα για τα στοιχεία που περιλαμβάνουν, όπως για παράδειγμα πίνακες, οι οποίοι είναι οι δομικές μονάδες για ένα παραδοσιακό σχεσιακό σύστημα, ούτε χρησιμοποιούν κάποια Structured Query Language (SQL) για την διαχείριση των δεδομένων. Χρησιμοποιούν αποκλειστικά non-relational τρόπους οργάνωσης και ανάλυσης των δεδομένων και είναι κατά κύριο λόγο βελτιστοποιημένες, ώστε να επισυνάπτουν και να ανακτούν τα δεδομένα αυτά . Τα κύρια χαρακτηριστικά τους συνοψίζονται στα εξής :

1) Η μη χρήση της SQL ως query language,
2) δεν μπορούν να εγγυηθούν ότι οι διεργασίες στην βάση δεδομένων θα γίνονται αξιόπιστα
3) έχουν μια ιδιαίτερη αρχιτεκτονική και φιλοσοφία λειτουργίας.

Το NoSQL κίνημα υιοθετήθηκε από τις μεγαλύτερες εταιρείες στον χώρο του Internet, όπως η Google, η Amazon και το Facebook. Και στις 3 αυτές περιπτώσεις προέκυψαν προκλήσεις στον χειρισμό τεράστιου όγκου δεδομένων, όπου οι συμβατικές RDBMS λύσεις, δεν μπορούσαν να ανταπεξέλθουν [19]. Και αυτό είναι το βασικό ατού των NoSQL συστημάτων. Έχουν την ικανότητα να αποθηκεύουν και να ανακτούν μεγάλες ποσότητες δεδομένων, «αδιαφορώντας» για τις σχέσεις μεταξύ των στοιχείων αυτών. Επίσης, οι μέθοδοι υλοποίησης και εφαρμογής τους αξιοποιούν μια αρχιτεκτονική που επιτρέπει (και ίσως

διευκολύνει) την κατανεμημένη λειτουργία του συστήματος. Τα χαρακτηριστικά αυτά οδηγούν το σύστημα σε αυξημένες επιδόσεις, αφού παρέχεται η δυνατότητα κλιμάκωσης (θεωρητικά άπειροι servers όπου κατανεμημένα θα επεξεργάζονται τα δεδομένα του συστήματος).Η σημαντική αυτή αύξηση στην απόδοση και την επεκτασιμότητα για ορισμένα μοντέλα δεδομένων, αντισταθμίζει την μειωμένη ευελιξία του χρόνου εκτέλεσης σε σύγκριση με συστήματα SQL (RDBMS).

Μπορούν να υποστηρίξουν πολλαπλές δραστηριότητες, συμπεριλαμβανομένων διαφόρων αναγνωριστικών και προγνωστικών analytics, την μετατροπή δεδομένων στυλ ETL, και μη κρίσιμες OLTP (για παράδειγμα, την διαχείριση μακράς διάρκειας ή ενδοεπιχειρηματικών συναλλαγών). Αρχικά τα συστήματα αυτά υποκινήθηκαν από Web 2.0 εφαρμογές και είναι σχεδιασμένα για αυξανόμενη κλιμάκωση (scaling) σε χιλιάδες ή εκατομμύρια χρήστες που κάνουν updates καθώς και reads, σε αντίθεση με τα παραδοσιακά DBMS συστήματα και τα data warehouses [20].

Η ιδέα είναι ότι και οι 2 τεχνολογίες μπορούν να συνυπάρξουν και η κάθεμία έχει την δική της θέση. Το κίνημα των NoSQL έχει αναδειχθεί τα τελευταία χρόνια καθώς πολύ πρωτοπόροι του Web 2.0, έχουν υιοθετήσει τα συστήματα αυτά. Εταιρείες όπως το Facebook, Twitter, Digg, Amazon, LinkedIn και η Google, όλες χρησιμοποιούν NoSQL με τον ένα ή με τον άλλο τρόπο.

2.2.2. Γιατί, που και πότε είναι χρήσιμες

Η συχνή πλέον χρήση μη-σχεσιακών βάσεων δεδομένων στον τομέα της τεχνολογίας, θα έκανε κάποιον να σκεφτεί ότι τα NoSQLσυστήματα έχουν αρχίσει να εκτοπίζουν τα παραδοσιακά σχεσιακά συστήματα. Αυτό στην πραγματικότητα δεν ισχύει. Κάθε ένα από αυτά τα δύο μοντέλα έχει διακριτό ρόλο και είναι κατάλληλο για διαφορετικές εφαρμογές και διαφορετικό φόρτο εργασίας. Στο κεφάλαιο αυτό θα επικεντρωθούμε στο γιατί να επιλέξει κάποιος να πάει σε ένα NoSQLσύστημα, πάντα σε σύγκριση με ένα SQL. Τα NoSQL συστήματα δεν εμφανίστηκαν για να αντικαταστήσουν τα σχεσιακά, αλλά για να τα συμπληρώσουν.

Τα NoSQL συστήματα διαχείρισης βάσεων δεδομένων είναι εξαιρετικά χρήσιμα όταν κάποιος δουλεύει με τεράστιο όγκο δεδομένων, η φύση των οποίων δεν απαιτεί κάποιο σχεσιακό μοντέλο. Για παράδειγμα εταιρείες που συλέγουν μεγάλες ποσότητες «αδόμητων» (unstructured) δεδομένων, στρέφονται όλο και περισσότερο σε μη-σχεσιακές βάσεις. Η κατανεμημένη τους φύση τα καθιστά ιδανικά για μαζική επεξεργασία δεδομένων (πχ ενοποίηση, φιλτράρισμα, διαλογή, στατιστικές ενέργειες κλπ). Είναι επίσης πολύ καλά

για ανάκτηση και ανταλλαγή δεδομένων μεταξύ μηχανημάτων (machine-to-machine), καθώς και για την επεξεργασία συναλλαγών μεγάλου όγκου, με την προυπόθεση βέβαια χαμηλών απαιτήσεων για ACID ιδιότητες.

Ως εκ τούτου, παρέχουν σχετικά φθηνή, υψηλής επεκτασιμότητας αποθήκευση μεγάλου όγκου, όπως για παράδειγμα ιστορικά δεδομένα, αρχεία καταγραφής, αρχεία τηλεφωνικών δεδομένων, ενδείξεις μετρητών και αισθητήρων, καθώς και αποθήκευση ημιδομημένων (semi-structured) ή αδόμητων δεδομένων (unstructured), όπως αρχεία ηλεκτρονικού ταχυδρομείου, αρχεία XML, έγγραφα, κλπ. Έχουν την ικανότητα της κλιμάκωσης, κάτι στο οποίο υστερούν τα παραδοσιακά συστήματα. Πέρα από τα πλεονεκτήματα κλίμακας, η ίδια η αρχιτεκτονική τους βοηθά στην βελτίωση της απόδοσής τους. Εαν μια σχεσιακή βάση δεδομένων έχει εκατοντάδες χιλιάδες πίνακες, η επεξεργασία των δεδομένων που βρίσκονται στους πίνακες αυτούς θα δημιουργήσει πολλά Locks στα δεδομένα, γεγονός το οποίο θα έχει σαν συνέπεια την υποβάθμιση της απόδοσης του συστήματος. Εν αντιθέση, τα NoSQL συστήματα έχουν πιο αδύναμα μοντέλα συνέπειας των δεδομένων, μπορούν να «θυσιάσουν» την συνοχή για την αποδοτικότητα [21].

Οι NoSQL λύσεις είναι ελκυστικές επειδή μπορούν να διαχειριστούν τεράστιες ποσότητες δεδομένων, σχετικά γρήγορα, σε ένα cluster από servers, οι οποίοι μοιράζονται πόρους μεταξύ τους. Επιπλέον, οι περισσότερες NoSQL λύσεις είναι ανοιχτού κώδικα (open source), κάτι το οποίο τους δίνει πλεονέκτημα τιμής έναντι των συμβατικών εμπορικών βάσεων δεδομένων.

2.2.3. Τι προσφέρουν και τι όχι

Αφού είδαμε πιο γενικά κάποια στοιχεία σχετικά με την λειτουργία των NoSQL συστημάτων και ποιες οι βασικές διαφορές τους από τα κλασικά συστήματα, τώρα θα αναφερθούμε πιο ειδικά στα πλεονεκτήματα τους, σε πιο πρακτικό επίπεδο. Οι NoSQL βάσεις δεδομένων επικεντρώνονται στην αναλυτική επεξεργασία **μεγάλης κλίμακας συνόλου δεδομένων (bigdata)**, προσφέροντας **αυξημένη επεκτασιμότητα και υψηλές επιδόσεις** [22]. Παρουσιάζουν μεγάλη «ελαστικότητα» στην βελτίωση της επίδοσης τους. Σε αυτό συμβάλλει η υλοποίησή τους, που τις περισσότερες φορές γίνεται σε υποδομές cloud ή σε virtualized environments, σε αντίθεση με τα κλασικά RDBMS συστήματα. Ειδικότερα, ενώ στα RDBMS συστήματα, όταν θέλουμε να τα βελτιώσουμε, προσθέτουμε περισσότερη μνήμη RAM ή καλύτερους επεξεργαστές στις υποδομές, αντίθετα στα NoSQL συστήματα απλά προσθέτουμε κόμβους ώστε να επεξεργάζονται ακόμη περισσότερα δεδομένα ταυτόχρονα. Ένα από τα μεγαλύτερα συν που έχουν είναι ότι υποστηρίζουν την

αποθήκευση και επεξεργασία μεγάλων όγκων δεδομένων (αδόμητα, ημι-δομημένα κλπ), κάτι στο οποίο υστερούν τα παραδοσιακα συστήματα, και κάτι το οποίο έχει γίνει αναπόφεκτο κομμάτι της σύγχρονης εποχής.

Βασικό πλεονέκτημα των μη-σχεσιακών συστημάτων είναι και ο **οικονομικός παράγοντας**. Για παράδειγμα, η συντήρηση ενός high-end συστήματος RDBMS που διαχειρίζεται τεράστιο όγκο δεδομένων απαιτεί εξίσου μεγάλο κόστος (κυρίως Database Administrators). Αντίθετα, τα NoSQL συστήματα εξ'ορισμού είναι σχεδιασμένα ώστε να απαιτούν τη λιγότερη δυνατή διαχείριση από τον ανθρώπινο παράγοντα. Αυτό επιτυγχάνεται χρησιμοποιώντας μεθόδους όπως το automatic repair, data distribution και πιο απλά μοντέλα δεδομένων. Όλα αυτά οδηγούν σε σαφώς μικρότερες ανάγκες για διαχείριση ή βελτιστοποίηση του συστήματος. Επιπλέον, οποιαδήποτε **αλλαγή στο μοντέλο δεδομένων** (schema) ενός συστήματος RDBMS είναι μία διαδικασία που απαιτεί ένα σεβαστό χρονικό διάστημα που η εφαρμογή που στηρίζεται στην συγκεκριμένη βάση δεδομένων δεν θα είναι προσβάσιμη (downtime) ή θα μειωθούν σε μεγάλο βαθμό τα επίπεδα λειτουργίας της υπηρεσίας, ενώ ένα NoSQL σύστημα έχει πολύ λιγότερους (έως μηδαμινούς) περιορισμούς σε αυτό το ζήτημα.

Από την άλλη μεριά βέβαια, τα NoSQL συστήματα **δεν μπορούν να εγγυηθούν ιδιότητες ACID**, σε σχέση με τις SQL βάσεις [23]. Στην πλειοψηφία τους "διακυνδυνεύουν" την συνοχή- παρουσιάζουν κάτι που γενικά ανεφέρεται ως eventual consistency. Για να πετύχουν υψηλή διαθεσιμότητα, θυσιάζουν την συνέπεια. Με ποιο απλά λόγια γενικά θεωρείται πως η ισχυρή συνέπεια είναι σύγχρονη. Μια δοσοληψία δεν θεωρείται ολοκληρωμένη παρά μόνο αν ενημερωθούν όλα τα αντίγραφα της βάσης δεδομένων. Η Eventual consistency γενικά θεωρείται ασύγχρονη. Τα αντίγραφα ενημερώνονται στο παρασκήνιο, χωρίς να μπλοκάρουν την ολοκλήρωση της δοσοληψίας. Μετά από κάποια αστοχία, μπορεί να υπάρχουν στο σύστημα πολλαπλές εκδόσεις ενός αντικειμένου, ασυνεπείς μεταξύ τους.

Ένα δεύτερο μειονέκτημα είναι ότι λόγω του «νεαρού της ηλικίας» τους και του ανοιχτού τους κώδικα, δεν προσφέρουν την **ασφάλεια υποστήριξης** και την **αξιοπιστία** που παρέχουν τα RDBMS συστήματα. Όπως έχει αναφερθεί νωρίτερα οι NoSQL βάσεις δεν χρησιμοποιούν σαν γλώσσα επερωτήσεων την SQL , αλλά πιο χαμηλές σε επίπεδο γλώσσες. Αυτό έχει σαν αποτέλεσμα να δημιουργούνται **περιορισμοί στην ευκολία ανάλυσης** των δεδομένων (δεν υπάρχουν joins και δεν υποστηρίζονται πάντα τα ad-hoc queries).

Στην επόμενη λίστα αναφέρονται επιγραμματικά τα βασικά πλεονεκτηματα και μειωνεκτήματα των NoSQL συστημάτων [24]:

- Είναι κατά βάση ανοικτού κώδικα
- Προσφέρουν οριζόντια επεκτασιμότητα
- Υποστηρίζουν Map/Reduce
- Είναι απλές στην χρήση
- Παρέχουν μεγάλες ταχύτητες στην εισαγωγή δεδομένων και στις απλές λειτουργίες επεξεργασίας
- Η δομή των δεδομένων (schema) μπορεί να αλλάξει χωρίς να επηρεαστεί τίποτα
- Έχουν την ικανότητα να αποθηκεύουν περίπλοκα είδη δεδομένων
- Θέματα διαχείρησης
- Δεν προσφέρουν δυνατότητα indexing
- Δεν παρέχουν λειτουργίες που βασίζονται στις ιδιότητες ACID
- Περίπλοκα μοντέλα συνέπειας/συνοχής (eventualconsistency)
- Έλλειψη τυποποίησης (σε επίπεδο API και querylanguage)

2.2.4. Κατηγορίες NoSQL συστημάτων

Τα NoSQL συστήματα μπορούν να κατηγοριοποιηθούν ως εξής με βάση την αρχιτεκτονική τους και στο μοντέλο δεδομένων που ακολουθούν [19]:

1) **Key-values Stores**: DynamoDB, FoundationDB, MemcacheDB, Redis, Riak, c-treeACE, Aerospike , Azure Table Storage, LevelDB, Berkeley DB, Oracle NOSQL Database, GenieDB, BangDB, Chordless, Scalaris, Tokyo Cabinet / Tyrant, Voldemort, MemcacheDB [19, 25]

2) **Column Family Stores**: Accumulo, Cassandra, Druid, HBase, Hypertable, Amazon SimpleDB, Cloudata, MonetDB [19, 25]

3) **Document Databases**: Clusterpoint, Apache CouchDB, Couchbase, MarkLogic, MongoDB, Elasticsearch, RethinkDB, NeDB, Terrastore,XML-dbs (BaseX, eXist, Sedna, Qizx) [19, 25]

4) **Graph Databases**: Allegro, Neo4J, InfiniteGraph, OrientDB, Virtuoso, Stardog, Sparksee, TITAN, InfoGrid, HyperGraphDB, GraphBase [19, 25]

Η πρώτη κατηγορία, οι key-value stores, βασίζονται στην ύπαρξη ενός hashtable όπου υπάρχει ένα μοναδικό κλειδί (key) και ένας δείκτης (pointer) που «δείχνει» σε ένα συγκεκριμένο στοιχείο. Είναι το πιο σχετικά απλό και εύκολο στην υλοποίηση μοντέλο από

τα NoSQL. Το μοντέλο αυτό συνοδεύεται από μηχανισμούς content caching, για την καλύτερη απόδοση του συστήματος σε μεγάλους όγκους δεδομένων [26].

Στις Column family stores τα δεδομένα αποθηκεύονται σε κελιά (cells), τα οποία ομαδοποιούνται σε στήλες (columns) και όχι σε σειρές (rows) όπως στις σχεσιακές βάσεις δεδομένων. Οι στήλες αυτές με την σειρά τους ομαδοποιούνται σε οικογένειες στηλών (column families), οι οποίες μπορούν να περιλαμβάνουν θεωρητικά άπειρο αριθμό από στήλες. Με τον τρόπο αυτό επιτυγχάνονται μεγάλες ταχύτητες σε λειτουργίες search/access, λόγω του ότι όλα τα κελιά που αναφέρονται σε μια στήλη είναι μια συνεχόμενη εγγραφή στο δίσκο [27].

Η επόμενη κατηγορία, των Document databases, ακολουθούν την λογική των key-value stores. Τα δεδομένα σε αυτή την περίπτωση (συλλογές από ζευγάρια key-value) είναι οργανωμένα, πιο φυσικά και λογικά, χωρίς να υπάρχουν περιορισμοί από κάποιο σχήμα. Κάθε εγγραφή (record) και τα συσχετιζόμενα μαζί της δεδομένα, θεωρούνται ως ένα document [28].

Η τελευταία κατηγορία, αυτή των Graph databases, χρησιμοποιεί δομές γράφων και είναι βασισμένη σε κόμβους (nodes), τις σχέσεις μεταξύ αυτών των κόμβων και τις ιδιότητές τους. Αντί για πίνακες με στήλες και σειρές, εδώ υπάρχει ένα ευέλικτο γραφικό μοντέλο (graphmodel) που μπορεί να χρησιμοποιηθεί και να αναπτυχθεί παράλληλα σε πολλά μηχανήματα (servers – κόμβους) [29].

Ακολουθεί πίνακας που κατατάσει τις γενικότερες κατηγορίες των NoSQL βάσεων ανάλογα με κάποια χαρακτηριστικά τους [30]:

Πίνακας 1: Μοντέλα δεδομένων και τα χαρακτηριστικά τους

Μοντέλο Δεδομένων	Απόδοση	Επεκτασιμότητα	Ευελιξία	Πολυπλοκότητα	Λειτουργικότητα
Key-value	high	high	high	none	Variable (none)
Column	high	high	moderate	low	minimal
Document	high	Variable (high)	high	low	Variable (low)
Graph	variable	variable	high	high	graph theory
Relational	variable	variable	low	moderate	relational algebra

Αφού αναφερθήκαμε στις κατηγορίες των NoSQL συστημάτων, θα κάνουμε μια πολύ σύντομη αναφορά στις πιο γνωστές και διαδεδομένες βάσεις δεδομένων NoSQL:

1) **Cassandra**: Η Cassandra είχε δημιουργηθεί από το Facebook και ανήκει πια στα Apache projects. Είναι μια column oriented NoSQL database και είναι ανοικτού κώδικα.

2) **Dynamo** και **SimpleDB**: Αποτελούν δημιουργίες της Amazon και η πρώτη είναι η πιο γνωστή Key-Value NoSQL βάση δεδομένων, ενώ η δεύτερη είναι μέρος των Amazon Cloud Services EC2 και S3.

3) **BigTable**: είναι δημιούργημα της Google, και η χρήση του περιορίζεται μόνο μέσω του Google App Engine. Τεχνικά, είναι μια column oriented database κλειστού κώδικα.

4) **Neo4J**: αποτελεί μια τύπου graph NoSQL database ελεύθερου κώδικα.

5) **CouchDB** και **MongoDB**: αυτές οι δύο αποτελούν τις πιο γνωστές open source document toriented NoSQL βάσεις.

Στα πλαίσια της εργασίας αυτής επιλέχθηκαν 3 NoSQl συστήματα για να μελετηθούν ποιοτικά και ποσοτικά, από ξεχωριστές κατηγορίες: η **Cassandra** από την Column Oriented Family και η **BaseX** με την **Mongo** από τις Document Oriented βάσεις. Στα επόμενα κεφάλαια θα περιγραφούν αναλυτικά οι βάσεις αυτές και θα γίνει ο ποιοτικός (Cassandra και BaseX) και ποσοτικός τους έλεγχος (Cassandra και Mongo).

2.3. Cassandra

Εικόνα3: Cassandra db http://cassandra.apache.org/]

Η Cassandra της Apache, είναι μία ανοιχτού κώδικα NoSQL βάση δεδομένων, η οποία είναι μαζικά επεκτάσιμη. Για το λόγο αυτό είναι ιδανική για να διαχειρίζεται τεράστιες ποσότητες δεδομένων σε διάφορα datacenters καθώς και στο cloud. Παρέχει διαρκή διαθεσιμότητα, γραμμική επεκτασιμότητα και λειτουργική απλότητα πάνω σε πολλούς servers χωρίς κανένα σημείο αποτυχίας (no single point of failure). Το μοντέλο δεδομένων που ακολουθεί είναι σχεδιασμένο για την μέγιστη ελαστικότητα και για πολύ γρήγορους χρόνους απόκρισης (response times) [31]. Αρχικά δημιουργήθηκε για το Facebook και είναι σχεδιασμένη να έχει συμμετρικούς Peer-to-peer κόμβους, αντί για master και named

κόμβους, ώστε να διασφαλίζει μηδενικό σημείο αποτυχίας (SPoF – single point of failure). Τα δεδομένα διαμοιράζονται αυτόματα σε όλους τους κόμβους του cluster, αλλά ο διαχειριστής μπορεί να καθορίσει ποια δεδομένα θα αντιγραφούν και πόσα αντίγραφα θα δημιουργηθούν [32]. Το 2009 η Cassandra μπήκε στα πλαίσια του Apache Incubator project, όταν το Facebook «άνοιξε» τον κώδικα της και το 2010 αναδείχθηκε ως top-level Apache project. Πήρε το όνομά της από μια μάντισσα στην αρχαία Ελληνική μυθολογία, της οποίας οι προβλέψεις δεν γίνονταν πιστευτές από τον κόσμο. Η Cassandra έχει υιοθετηθεί και χρησιμοποιείται αυτή τη στιγμή από τεράστιους οργανισμούς και εταιρείες όπως : Netflix, Digg, Adobe, Twitter, HP, IBM, Rackspace, Cisco, Reddit [32] κλπ.

Ας δούμε τώρα κάπως περιληπτικά σε ποια σημεία διαφέρει η Cassandra από μια απλή σχεσιακή βάση δεδομένων [33]:

Πίνακας 2: Cassandra vs Σχεσιακών βάσεων

Σχεσιακή Βάση	Cassandra
Διαχειρίζεται μέτρια ταχύτητα εισερχόμενων δεδομένων	Διαχειρίζεται υψηλή ταχύτητα εισερχόμενων δεδομένων
Δεδομένα έρχονται από μια ή λίγες τοποθεσίες	Δεδομένα έρχονται από πολλές τοποθεσίες
Διαχειρίζεται κυρίως δομημένα δεδομένα (structured)	Διαχειρίζεται όλα τα είδη των δεδομένων
Υποστηρίζει πολύπλοκες συναλλαγές	Υποστηρίζει πολύ απλές συναλλαγές
Υπάρχουν σημεία αποτυχίας (failover)	Κανένα σημείο αποτυχίας (συνεχές κ σταθερό uptime)
Υποστηρίζει μέτριο όγκο δεδομένων	Υποστηρίζει τεράστιο όγκο δεδομένων
Centralized deployments	decentralized deployments
Τα δεδομένα είναι γραμμένα κυρίως σε μια τοποθεσία	Τα δεδομένα είναι γραμμένα σε πολλαπλές τοποθεσίες
Υποστηρίζει επεκτασιμότητα στο read (θυσιάζοντας την συνοχή/συνέπεια)	Υποστηρίζει επεκτασιμότητα σε read και write
Αναπτύσσεται σε κατακόρυφη κλιμάκωση (vertical scaleup)	Αναπτύσσεται σε οριζόντια κλιμάκωση (horizontal scaleout)

2.3.1. Αρχιτεκτονική και Datamodel

Η Cassandra είναι βασικά ένας συνδιασμός του Bigtable της Google και του Dynamo της Amazon. Έχει πολλά περισσότερα χαρακτηριστικά από ένα σύστημα τύπου key/value, όπως η Riak, αλλά υποστηρίζει λιγότερα είδη επερωτημάτων σε σχέση με μια document-based βάση δεδομένων, όπως η Mongo [34]. Ο σχεδιασμός της βασίστηκε στην φιλοσοφία του ότι αστοχίες συστήματος/hardware πάντα συμβαίνουν, και έτσι με βάση τη λογική αυτή σχεδιάστηκε ως ένα peer-to-peer κατανεμημένο σύστημα. Όλοι οι κόμβοι είναι ιεραρχικά

ίδιοι (απουσιάζει το μοντέλο του master και των name nodes). Τα δεδομένα είναι κατανεμημένα σε όλους τους κόμβους του cluster και αντιγράφονται και μοιράζονται αυτόματα και με διαφάνεια, ώστε να διασφαλίζεται η ανοχή σε τυχόν βλάβες/αστοχίες του συστήματος [35].

Προσφέρει επίσης προσαρμοσμένη αντιγραφή, αποθηκεύει δηλαδή αντίγραφα των δεδομένων σε όλους τους κόμβους που συμμετέχουν σε ένα Cassandra ring.Αυτό σημαίνει πως αν κάποιος κόμβος βγει εκτός λειτουργίας, ένα ή περισσότερα αντίγραφα των δεδομένων που είχε αυτός ο κόμβος, θα υπάρχουν και θα είναι διαθέσιμα σε κάποιον άλλο κόμβο του cluster [36]. Χρησιμοποιεί consistent hashing για να αναθέσει δεδομένα στους κόμβους. Με απλά λόγια, η Cassandra χρησιμοποιεί έναν hash αλγόριθμο για να υπολογίσει το hash για τα κλειδιά κάθε αντικειμένου που είναι αποθηκευμένο στη βάση (πχ.το όνομα της στήλης, το ID της γραμμής). Το εύρος των τιμών του hash (αλλιώς το keyspace) διαιρείται ανάμεσα στους κόμβους του cluster. Έπειτα η Cassandra αντιστοιχεί κάθε δεδομένο σε έναν κόμβο, και αυτός ο κόμβος είναι υπεύθυνος για την αποθήκευση και την διαχείριση τους συγκεκριμένου δεδομένου[37].

Η Cassandra παρέχει δυνατότητα γραμμικής κλιμάκωσης. Η δυναμική του συστήματος μπορεί πολύ εύκολα να αυξηθεί απλά προσθέτοντας νέους κόμβους στο δίκτυο. Για παράδειγμα, αν 2 κόμβοι μπορούν να διαχειρίζονται 100,000 λειτουργίες ανά δευτερόλεπτο, 4 κόμβοι μπορούν να υποστηρίξουν 200,000 λειτουργίες ανά δευτερόλεπτο, ενά 8 κόμβοι 400,000 λειτουργίες ανά δευτερόλεπτο [36]:

Εικόνα 4: Γραμμική κλιμάκωση στην Cassandra [http://www.datastax.com/documentation/cassandra/2.0/cassandra/gettingStartedCassandraIntro.html]

Λόγω του peer-to-peer σχεδιασμού έχει αρχιτεκτονικό στυλ τύπου Read/Write –anywhere. Εγγραφή και ανάκτηση μπορεί να γίνει προς και από οποιοδήποτε κόμβο. Κάθε κόμβος επικοινωνεί με τον άλλο μέσω του πρωτοκόλου Gossip, το οποίο ανταλλάσει πληροφορίες μέσα στο cluster κάθε δευτερόλεπτο. Οι πληροφορίες αφορούν ποιοι κόμβοι λειτουργούν,

ποιοι όχι, και γενικότερα την δραστηριότητα μέσα στο cluster. Όταν γίνεται εγγραφή δεδομένων αυτά γράφονται πρώτα σε ένα Commit log, ώστε να διασφαλιστεί η ανθεκτικότητα τους (data durability). Στη συνέχεια τα δεδομένα γράφονται σε μια δομή εντός της μνήμης, το memtable (κάτι σαν το write-back cache). Όταν αυτό γεμίσει, τα δεδομένα περνάνε στον δίσκο σε μια άλλη δομή που ονομάζεται SStable (Shorten String table)[35]. Όλες οι εγγραφές αντιγράφονται και κατανέμονται αυτόματα μέσα στο cluster. Χρησιμοποιώντας περιοδικά μια διαδικασία που ονομάζεται «συμπύκνωση» (compaction), ενοποιεί τα SSTables, απορρίπτοντας τα άχρηστα δεδομένα ή αυτά που έχουν διαγραφεί [36].

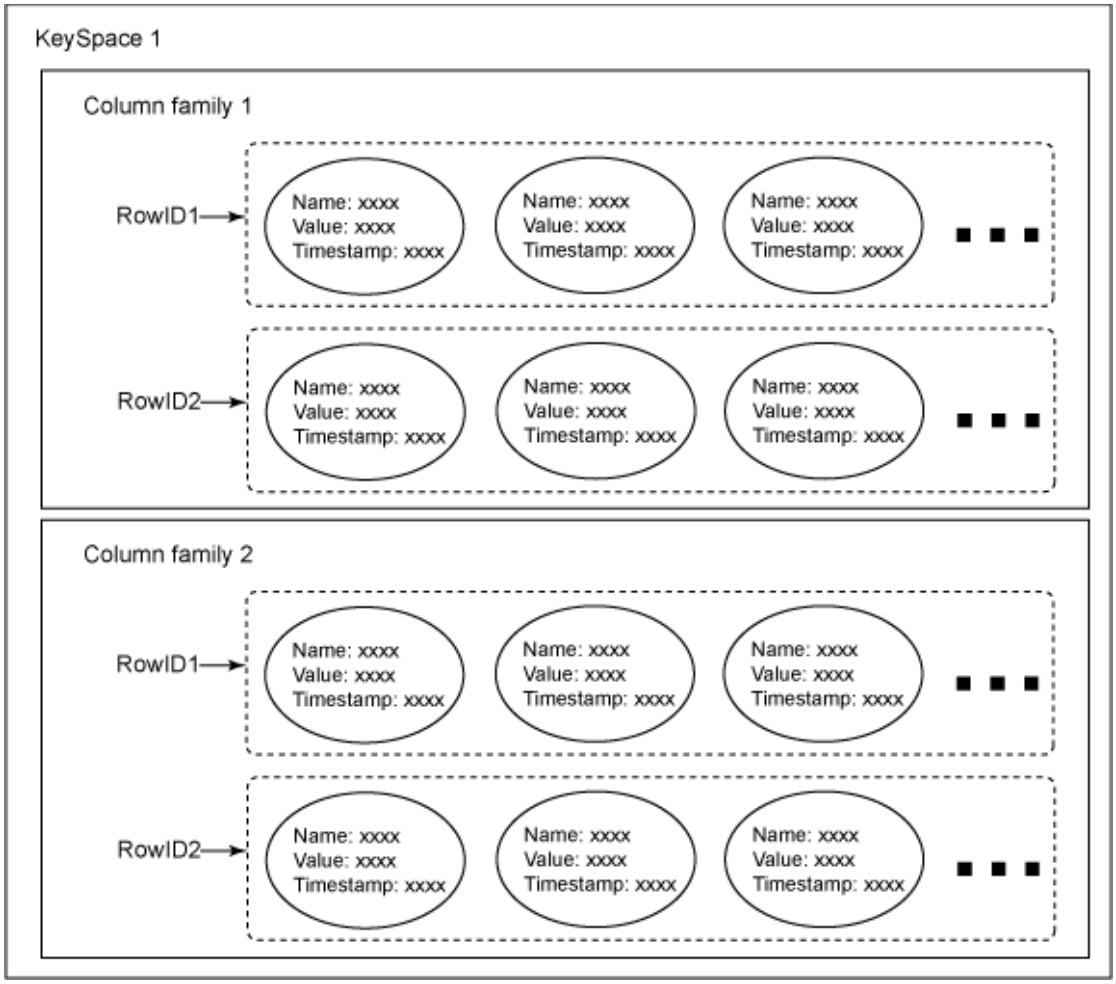
Η αρχιτεκτονική της είναι φτιαγμένη ώστε να υποστηρίζει πολύ γρήγορα Writes (συγκριτικά με τα Reads). Αυτό επιτυγχάνεται με τις εξής 2 μεθόδους:

1. Κρατάει τα περισσότερα δεδομένα μέσα στην μνήμη του υπεύθυνου κόμβου και όλα τα updates γίνονται μέσα στην μνήμη και γράφονται στο filesystem με ένα «χαλαρό» τρόπο. Για να αποφύγει την απώλεια δεδομένων, γράφει όλες τις συναλλαγές σε ένα commit log στον δίσκο. Οι εγγραφές στο commitlog είναι append-only (εν αντιθέση με τα updates στον δίσκο) και έτσι γλυτώνει καθυστέρηση περιστροφής κατά την εγγραφή στον δίσκο [37].

2. Αν οι εγγραφές δεν ζητήσουν πλήρη συνέπεια, η Cassandra γράφει τα δεδομένα σε αρκετούς κόμβους χωρίς να δίνει σημασία σε τυχόν ασυνέπειες αυτών. Επιλύει τις ασυνέπειες στην συνέχεια κατά το πρώτο διάβασμα. Η διαδικασία αυτή ονομάζεται "read repair"[37].

Οι αιτήσεις εγγραφής ή ανάκτησης (write και read) ενός χρήστη μπορούν να σταλούν σε οποιονδήποτε κόμβο του cluster. Όταν ο χρήστης συνδεθεί σε έναν κόμβο με ένα αίτημα, ο κόμβος λειτουργεί ως συντονιστής για το συγκεκριμένο αίτημα μεταξύ της εφαρμογής του χρήστη και των κόμβων που έχουν τα δεδομένα προς ζήτηση. Εκείνος καθορίζει ποιοί κόμβοι του συστήματος θα πάρουν το αίτημα ανάλογα με τον τρόπο που έχει στηθεί το cluster [36].

Το σχήμα (schema) που χρησιμοποιεί η Cassandra είναι εμπνευσμένο από το Bigtable της Google. Είναι ένα row-oriented, column structure design και εμπεριέχει τις εξής έννοιες/ επίπεδα [35] :



Εικόνα5: Cassandra data model [http://www.ibm.com/developerworks/library/os-apache-cassandra/]

Keyspace→ Ένα γκρουπ από πολλές οικογένειες στηλών. Είναι παρόμοιο με την έννοια της βάσης δεδομένων στον κόσμο των σχεσιακών συστημάτων και πρέπει να δηλώνεται από την αρχή. Είναι το εξωτερικό επίπεδο του συστήματος και είναι συνήθως το όνομα της εφαρμογής. Για παράδειγμα το "Twitter" είναι ένα keyspace [34].

Column family→ είναι το βασικό αντικείμενο διαχείρισης των δεδομένων και είναι το αντίστοιχο ενός πίνακα στις σχεσιακές βάσεις, με τη διαφορά ότι το σχήμα είναι δυναμικό και εύκαμπτο. Κάθε οικογένεια στηλών αποθηκεύεται σε ένα ξεχωριστό αρχείο στο δίσκο. Και αυτές πρέπει να καθορίζονται από την αρχή [34]. Είναι ουσιαστικά μια συλλογή σειρών κάτω από ένα κοινό όνομα [37].

Row/Key→ μια σειρά σε μια οικογένεια στηλών χαρακτηρίζεται από ένα κλειδί (key), το οποίο είναι η μόνιμη ονομασία της εγγραφής. Είναι ουσιαστικά μια συλλογή από στήλες με κοινό όνομα. Μπορούν να υποβληθούν ερωτήματα πάνω σε σειρές κλειδιών μέσα σε μια οικογένεια στηλών [34].Η Cassandra αποτελείται από πολλούς αποθηκευτικούς κόμβους

και αποθηκεύει κάθε σειρά σε έναν κόμβο. Μέσα σε κάθε σειρά, αποθηκεύει πάντα στήλες με αλφαβητική σειρά. Χρησιμοποιώντας την σειρά αυτή, υποστηρίζει slice queries, όπου δεδομένης της σειράς, οι χρήστες μπορούν να ανακτήσουν ένα υποσύνολου των στηλών που εμπίπτει με το δωσμένο εύρος ονομάτων [37].

Μετά το επίπεδο της ***Column family,*** υπάρχουν δύο πιθανότητες επιλογής :

Η ***στήλη (Column = cell)*** →είναι η βασική μονάδα του data model και αποτελείται από ένα σετ ονόματος και τιμής (name/value).Οι στήλες ορίζονται on the fly και διαφορετικά αρχεία μπορούν να έχουν διαφορετικά σε από ονόματα στηλών, ακόμα και μέσα στο ίδιο ***Keyspace*** και ***Column family.*** Αυτό επιτρέπει το όνομα της στήλης να χρησιμοποιηθεί είτε ως δομή, είτε ως δεδομένο. Όλες οι στήλες κρατούν timestamp [34].

Η ***υπέρ-στήλη (Supercolumn)***→ είναι ουσιαστικά μια named list. Περιέχει κανονικές στήλες. Ορίζονται on the fly και μπορούν να είναι πάρα πολλές ανά σειρά. Αποθηκέυονται σε αλφαβητική σειρά, με τις υπο στήλες τους γειτονικά, σε φυσικό επίπεδο, στον δίσκο [34].

Γενικά οι κανονικές στήλες και οι υπερστήλες δεν πρέπει να συγχέονται στο ίδιο (4ο) επίπεδο διάστασης. Πρέπει να ορίζεται ρητά και εξ αρχής ποιες οικογένειες στηλών περιέχουν κανονικές στήλες και ποιες υπερ-στήλες, με κανονικές στήλες μέσα σε αυτές. Οι υπερ-στήλες προσθέτουν μια $5^{η}$ διάσταση στο υβριδικό μοντέλο της Cassandra, μετατρέποντας τις στήλες σε λίστες. Αυτό είναι πολύ χρήσιμο σε per-user indexes [34].

2.3.2. Σύγκριση Cassandra και RDBMS

Με βάση το μοντέλο δεδομένων που ακολουθεί η Cassandra, αυτά τοποθετούνται σε χώρο 2 διαστάσεων μέσα σε κάθε οικογένεια στηλών. Για να ανακτήσει ο χρήστης τα δεδομένα σε μιά οικογένεια στηλών, χρειάζεται δυο κλειδιά: το όνομα της σειράς και το όνομα της στήλης. Υπό αυτή την έννοια η Cassandra και το σχεσιακό μοντέλο είναι παρόμοια , αν και υπάρχουν αρκετές κρίσιμες διαφορές [37].

1. Οι σχεσιακές στήλες είναι ομοιογενείς σε όλες τις γραμμές του πίνακα. Μια καθαρά κάθετη σχέση υπάρχει συνήθως μεταξύ των δεδομένων. Δεν συμβαίνει το ίδιο όμως στην περίπτωση της Cassandra. Για το λόγο αυτό αποθηκεύει το όνομα της στήλης μαζί με κάθε στήλη [37].

2. Στο σχεσιακό μοντέλο κάθε σημείο του 2D χώρου πρέπει να έχει τουλάχιστον την κενή τιμή αποθηκευμένη εκεί, ώστε ο χώρος να είναι πλήρης. Στην Cassandra δεν συμβαίνει

αυτό καθώς μπορεί να έχει γραμμές που περιέχουν ελάχιστα αντικείμενα και άλλες που περιέχουν χιλιάδες [37].

3. Στο σχεσιακό μοντέλο το σχήμα είναι προκαθορισμένο και δεν μπορεί να αλλάξει κατά τον χρόνο εκτέλεσης. Εν αντιθέση η Cassandra επιτρέπει στον χρήστη να αλλάξει το σχήμα όποτε το επιθυμεί [37].

4. Η Cassandra αποθηκεύει πάντα τα δεδομένα με τρόπου τέτοιο ώστε οι στήλες να είναι με αλφαβητική σειρά. Αυτό κάνει πιο εύκολη την αναζήτηση δεδομένων μέσα σε μια στήλη χρησιμοποιώντας slice queries. Από την άλλη μεριά είναι αρκετά δυσκολότερο να ψάξεις μέσα σε μια σειρά, εκτός και αν χρησιμοποιείται κάποιος order-preserving partitioner [37].

5. Μια ακόμα κρίσιμη διαφορά είναι ότι τα ονόματα των στηλών σε μια σχεσιακή βάση αντιπροσωπεύουν μεταδεδομένα σχετικά με τα δεδομένα, αλλά ποτέ τα ίδια τα δεδομένα. Στην Cassandra τα ονόματα των στηλών μπορούν να περιλαμβάνουν δεδομένα. Συνεπώς, οι σειρές στην Cassandra μπορούν να έχουν χιλιάδες στήλες, ενώ σε μια σχεσιακή βάση συνήθως έχουν λίγες δεκάδες [37].

6. Χρησιμοποιώντας ένα καλά καθορισμένο, αμετάβλητο σχήμα, τα σχεσιακά μοντέλα υποστηρίζουν σύνθετα επερωτήματα που περιλαμβάνουν JOINs, συναθροίσεις και πολλά άλλα. Με το σχεσιακό μοντέλο ο χρήστης μπορεία να ορίσει το σχήμα των δεδομένων χωρίς να ανησυχεί για τα queries. Η Cassandra δεν υποστηρίζει JOINs, ούτε τις περισσότερες SQL μεθόδους αναζήτησης. Ως εκ τούτου, το σχήμα πρέπει να είναι προσαρμοσμένο στα queries που απαιτούνται από την εκάστοτε εφαρμογή [37]. Ένας τρόπος με τον οποίο αποφεύγονται τα joins είναι να αποθηκεύει τις πληροφορίες ενός συγκεκριμένου αντικειμένου μέσα στην ίδια στήλη. Με τον τρόπο αυτό ο χρήστης μπορεί να πάρει όλα τα δεδομένα για ένα συγκεκριμένο αντικείμενο με ένα μόνο query.

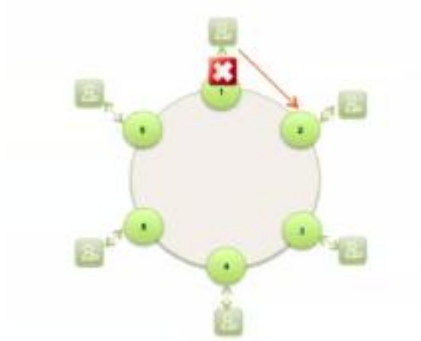
2.3.3.Πλεονεκτήματα και features της Cassandra

Είναι ιδανική βάση για web-scale domains όπως τα social networks. Ακολουθεί λίστα με τα βασικά χαρακτηριστικά, πλεονεκτήματα και μειονεκτήματά της.

1. **Υποστηρίζει μεγάλο όγκο δεδομένων→** Προσφέρει επεκτασιμότητα από Gigabyte σε Petabyte. Παρέχει κέρδη γραμμικής απόδοσης προσθέτοντας νέους κόμβους [35]. Για να μεγαλώσει η χωρητικότητα του cluster αρκεί να προστεθεί ένας ακόμη κόμβος, χωρίς να γίνει επανεκκίνηση διαδικασιών πόυ τρέχουν ή να μεταφερθούν δεδομένα χειροκίνητα [34].

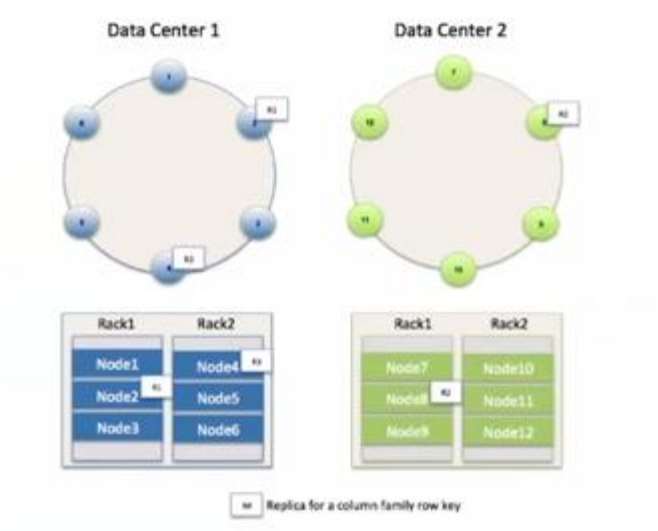
Εικόνα6: Throughput capabilities [from DataStax Cassandra Tutorials]

2. **Κανένα σημείο αποτυχίας (No single point of failure)**→Όλοι οι κόμβοι είναι το ίδιο σε θέμα ιεραρχίας και ο χρήστης μπορεί να γράψει ή να διαβάσει από οποιονδήποτε κόμβο. Παρέχει επίσης τη δυνατότητα αναπαραγωγής των δεδομένων μεταξύ διαφόρων φυσικών κέντρων (datacenters) [35].Κατανέμει τα δεδομένα στους κόμβους με διαφάνεια στους χρήστες. Κάθε κόμβος μπορεί να δεχτεί οποιοδήποτε αίτημα (read, write, ή delete) και να το προωθήσει στον κατάλληλο κόμβο εάν τα δεδομένα δεν ανήκουν σε αυτόν [37].



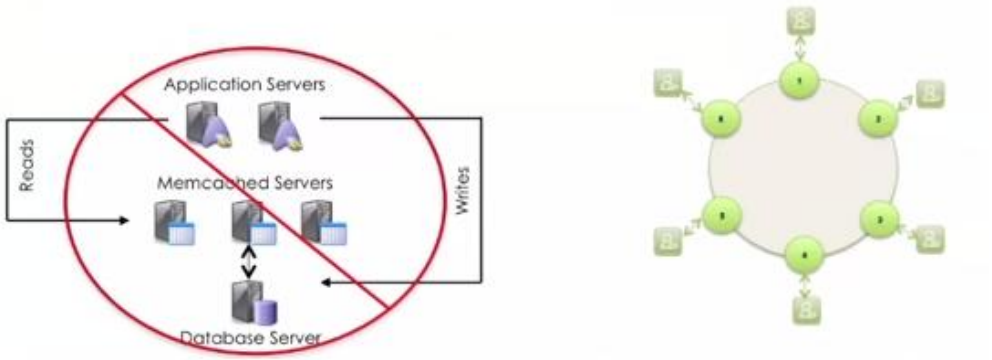
Εικόνα7: No single point of failure [from DataStax Cassandra Tutorials]

3. **Εύκολη αντιγραφή και διαμοιρασμός των δεδομένων**→ εκμεταλλεύεται όλα τα ωφέλη του cloud computing και προσφέρει δυνατότητα για multi-datacenter [35]. Προσφέρει επίσης multi-datacenter awareness. Ο χρήστης μπορεί να διαμορφώσει το layout ενός κόμβου, ώστε σε περίπτωση καταστροφής του κόμβου αυτού, να υπάρχει εναλλακτικό datacenter που θα περιέχει τουλάχιστον ένα πλήρες αντίγραφο της κάθε εγγραφής [34]. Ο χρήστης μπορεί να ορίσει πόσα αντίγραφα θα υπάρχουν και η Cassandra χειρίζεται την δημιουργία και την διαχείριση των αντιγράφων με διαφάνεια [37].



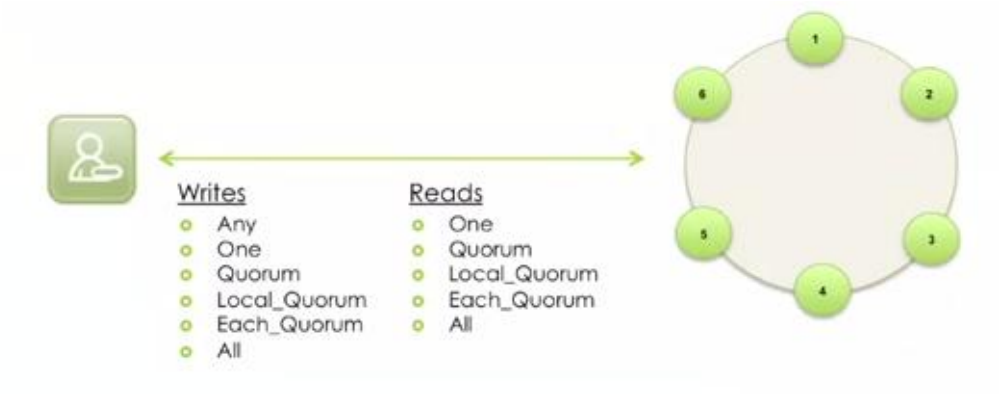
Εικόνα 8: αντιγραφή και διαμοιρασμός των δεδομένων [from DataStax Cassandra Tutorials]

4**. Δεν υπάρχει ανάγκη για ξεχωριστό caching layer**→Η αρχιτεκτονική Peer-to-peer αφαιρεί την ανάγκη για ειδικό caching layer. Το cluster της βάσης χρησιμοποιεί τη μνήμη όλων των κόμβων που συμμετέχουν για να κάνει cache στα δεδομένα που βρίσκονται σε κάθε κόμβο. Με τον τρόπο αυτό αποφεύγονται παρατυπίες μεταξύ της προσωρινής μνήμης και της βάσης [35].



Εικόνα9: αρχιτεκτονική Peer-to-peer [from DataStax Cassandra Tutorials]

5. **Ρυθμιζόμενη συνοχή δεδομένων**→ παρέχεται η επιλογή μεταξύ ισχυρής και ενδεχόμενης συνέπειας, ανάλογα με την ανάγκη. Μπορεί να γίνει και για λειτουργίες read και write [35]. Κατά την αποθήκευση και την ανάγνωση των δεδομένων, ο χρήστης μπορεί να επιλέξει το απαιτούμενο επίπεδο συνέπειας για κάθε λειτουργία. Για παράδειγμα όταν χρησιμοποείται το επίπεδο "quorum" κατά την εγγραφή ή την ανάγνωση, αυτό σημαίνει ότι τα δεδομένα γράφονται και διαβάζονται από πάνω από τους μισούς κόμβους του cluster. Η υποστήριξη ρυθμιζόμενης συνοχής δίνει την δυνατότητα στους χρήστες να επιλέγουν το επίπεδο που ταιριάζει στην κάθε ξεχωριστή περίπτωση χρήσης [37].



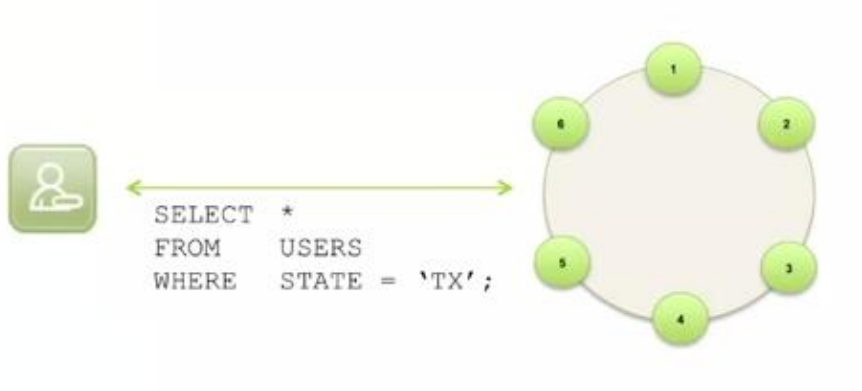
Εικόνα10: Data consistency [from DataStax Cassandra Tutorials]

6. **Ευέλικτο σχέδιο σχήματος→** ο δυναμικός σχεδιασμός του σχήματος επιτρέπει ευέλικτη αποθήκευση δεδομένων σε σχέση με το «απόλυτο» σχεσιακό μοντέλο. Δεν χρειάζεται να είναι προκαθορισμένα τα πεδία των δεδομένων, καθώς αυτά μπορούν να προστεθούν ή να αφαιρεθούν on the fly. Για μεγάλα deployments αυτό δίνει μεγάλη ώθηση στην παραγωγικότητα [34]. Μπορεί να διαχειρίζεται δομημένα, ημι-δομημένα και αδόμητα δεδομένα. Δεν υπάρχουν χρόνοι εκτός λειτουργίας για αλλαγές στο σχήμα. Υποστηρίζει Indexing πρωτου και δεύτερου βαθμού [35].

7. **Συμπίεση δεδομένων→** Χρησιμοποιεί τον αλγόριθμο συμπίεσης δεδομένων Snappy της Google και κάνει την συμπίεση σε επίπεδο οικογένειας στηλών. Συμπιέζει τα δεδομένα μέχρι και 80% [35].

Εικόνα 11: Συμπίεση Δεδομένων [fromDataStaxCassandraTutorials]

8. **Γλώσσα επερωτήσεων CQL (σαν την SQL)→** Το συντακτικό είναι παρόμοιο με το αντίστοιχο της SQL. Δημιουργεί objects μέσω DDL (πχ. CREATE...). Υποστηρίζονται οι βασικές εντολές: INSERT, UPDATE, DELETE, SELECT [35]. Υποστηρίζει range queries σε αντίθεση με τα περισσότερα key/value συστήματα. Παρέχει δηλαδή την δυνατότητα για επερωτήματα για διατεταγμένα εύρη κλειδιών (ordered ranges of keys)[34].



Εικόνα12: γλώσσα CQL [from DataStax Cassandra Tutorials]

9. Όπως προαναφέρθηκε η Cassandra αποθηκεύει τις στήλες σε σειρά με βάση το όνομά τους. Με τον τρόπο αυτό τα slice queries είναι πάρα πολύ γρήγορα. Αξίζει να σημειωθεί ότι η αποθήκευση όλων των λεπτομερειών ενός στοιχείου στην ίδια σειρά και η χρήση εντολών sort, είναι οι πιο σημαντικές ιδέες πίσω από τον σχεδιασμό της Cassandra (σε επίπεδο δεδομένων) [37].

10. **Πολύ γρήγορα writes**→στην πραγματικότητα είναι γρηγορότερα από τα reads (μεταφέρει δεδομένα με ταχύτητες 80-360 MB/sec ανά κόμβο).

2.3.4. Μειονεκτήματα της Cassandra

Αν και η αρχιτεκτονική της Cassandra είναι υψηλής επεκτασιμότητας, υπάρχουν πάντα κάποιες θυσίες που πρέπει να γίνουν στα κατανεμημένα συστήματα.

1. **Δεν υποστηρίζει συναλλαγές ACID→** αν και έχει μια λειτουργία batch, δεν υπάρχει καμία εγγύηση ότι οι υπο-λειτουργίες γίνονται σε ατομικό επίπεδο [37].

2**. Δεν υποστηρίζει JOINs→** εάν ο χρήστης θέλει να ενώσει 2 οικογένειες στηλών, πρέπει να ανακτήσει και να ενώσει τα δεδομένα με τρόπο προγραμματιστικό. Για μεγάλα datasets η διαδικασία αυτή είναι χρονοβόρα και ακριβή. Η Cassandra παρακάμπτει τον περιορισμό αυτό αποθηκεύοντας όσο περισσότερα δεδομένα γίνεται μέσα στην ίδια σειρά (row) [37].

3**. Δεν υποστηρίζει μεθόθους αναζήτησης εκτός βασικού σχεδιασμού**. Αν και υποστηρίζει secondary indexes, αυτοί έχουν πολλούς περιορισμούς, όπως δεν υποστηρίζουν επαρκώς range queries [37]. Έτσι για τα καλύτερα αποτελέσματα ο χρήστης πρέπει να εφαρμόζει αναζητήσεις χτίζοντας δικά του indexes και χρησιμοποιώντας εντολές ταξινόμησης στήλης και σειράς. Δεν μπορείς να κάνεις ad-hoc queries χωρίς να χτίσεις νέα indexes (περίπλοκο από την πλευρά των προγραμματιστών). Για κάθε νέο query χρειάζεται και ένα νέο index.

4. **Ιδιαιτερότητα με τα κλειδιά→** ο χρήστης δεν μπορεί να αλλάξει τα κλειδιά, τα οποία πρέπει να είναι και μοναδικά. Αν ένα κλειδί χρησιμοποιηθεί 2 φορές, τα δεδομένα θα γίνουν overwrite. Για το λόγο αυτό μπορούν να χρησιμοποιηθούν σύνθετα κλειδιά (συνήθως στα κλειδιά των σειρών). Ένας άλλος τρόπος (συνήθως στα κλειδιά των στηλών), είναι μια επιδιόρθωση του κλειδιού με μια τυχαία τιμή ή ένα timestamp [37].

5. **Αποτυχίες λειτουργιών→** όπως αναφέρθηκε στο 1, η Cassandra δεν υποστηρίζει ατομικές λειτουργίες αλλά λειτουργίες που αφήνουν το σύστημα στην ίδια κατάσταση ανεξάρτητα από τις φορές που εκτελείται μια λειτουργία (Idempotent operations). Αν κάποια από αυτές αποτύχει, ο χρήστης μπορεί να ξαναπροσπαθήσει όσες φορές θέλει χωρίς κάποιο πρόβλημα. Αυτό παρέχει έναν μηχανισμό ανάκτησης από παροδικές αποτυχίες. Αν μια λειτουργία αποτύχει ακόμα και μετά από κάποιες προσπάθειες, ενδέχεται να προκαλέσει «παρενέργειες» [37].

Οι περιορισμοί και τα μειονεκτήματα που παρουσιάστηκαν για την περίπτωση της Cassandra είναι συχνά σε όλες τα NoSQL συστήματα. Συχνά όμως τα προβλήματα αυτά παραβλέπονται και γίνονται συμβιβασμοί στο όνομα της υψηλής επεκτασιμότητας.

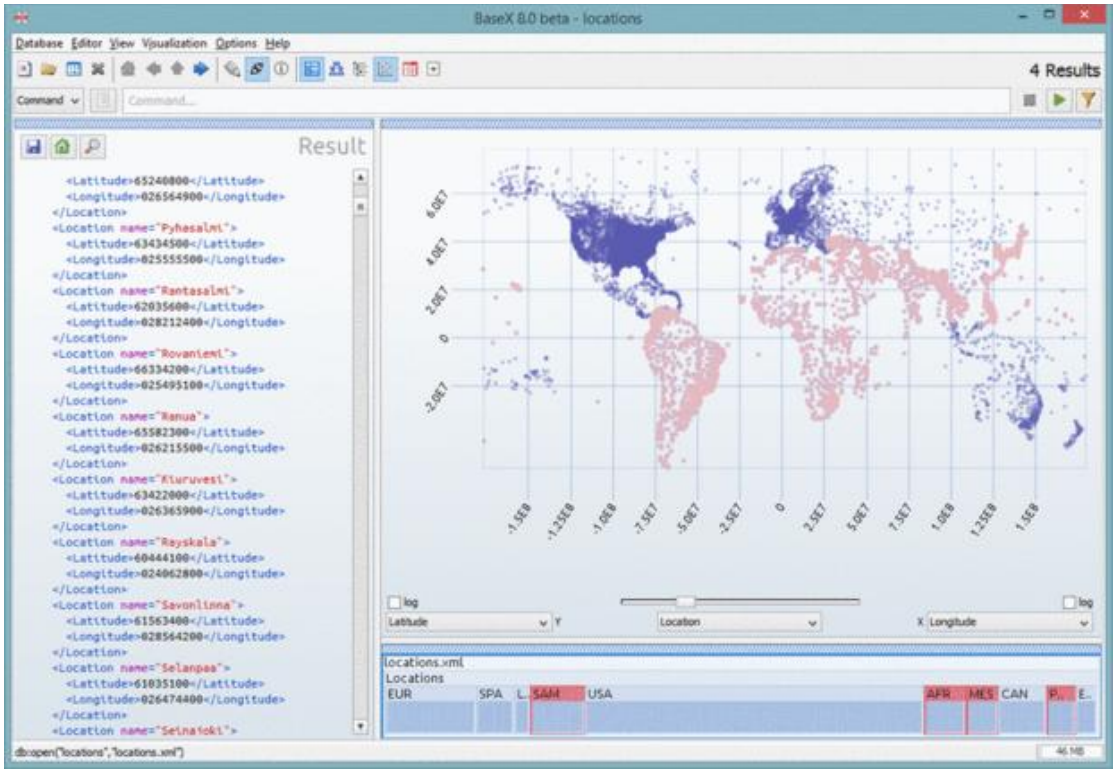
2.4. BaseX

Εικόνα13: BaseX db [basex.org]

Η BaseX ανήκει στην κατηγορία των document-oriented NoSQL συστημάτων, και συγκεκριμένα είναι μια ελαφριά XML βάση δεδομένων και ταυτόχρονα επεξεργαστής της XQuery. Αναπτύχθηκε ως community project στο GitHub [38]. Είναι ανεξάρτητη πλατφόρμα και διανέμεται κάτω από άδεια χρήσης ελεύθερου λογισμικού. Εξειδικεύεται στην αποθήκευση, την αναζήτηση και την οπτικοποίηση πολύ μεγάλων εγγράφων και συλλογών XML. Η BaseX σαν Project ξεκίνησε από τον Christian Grün του πανεπιστημίου του Konstanz το 2005. Το 2007 έγινε ανοιχτού κώδικα και από τότε κυκλοφορεί υπό την BSD άδεια.

Οι βάσεις δεδομένων που ανήκουν στην υποκατηγορία των XML βάσεων παρέχουν υποστήριξη για τυποποιημένες γλώσσες επερωτήσεων όπως η XPath και η XQuery, σε

αντίθεση με τις υπόλοιπες document-oriented βάσεις. Η BaseX είναι ελαφριά, υψηλής απόδοσης και επεκτασιμότητας και σύμφωνη με τις προδιαγραφές του World Wide Web Consortium (W3C) Update και τις FullText προεκτάσεις [39]. Το συμπεριλαμβανόμενο και φιλικό προς τον χρήστη GUI επιτρέπει στους χρήστες να αναζητούν, να εξερευνούν και να εναλύουν τα δεδομένα τους με έναν διαδραστικό τρόπο και να αξιολογούν τις εντολές XPath/Xquery σε πραγματικό χρόνο (την ώρα που ο χρήστης πληκτρολογεί). Στην παρακάτω εικόνα ακολουθεί ένα παράδειγμα του τι μπορεί να κάνει η BaseX. Εδώ απεικονίζεται το γραφικό της περιβάλλον όπου δείχνει σύνολα δεδομένων γεωγραφικής θέσης. Παρέχει ένα query interface και ταυτόχρονα μια γραφική απεικόνιση των δεδομένων, παίρνοντας το γεωγραφικό πλάτος και μήκος ως τιμές X και Y, και από αυτές παράγει έναν χάρτη. Τα δεδομένα επίσης μπορούν να εμφανιστούν με τη μορφή δέντρου και table views [40].



Εικόνα 14: Γραφική απεικόνιση δεδομένων σε χάρτη [http://www.networkworld.com/article/2892245/opensource-subnet/basex-free-open-source-xml-wrangling.html]

Οι τεχνολογίες που υποστηρίζει είναι οι εξής:

- Γλώσσα επερωτήσεων XPath
- XQuery 3.1
    - XQuery Update 3.0 (W3C)
    - XQuery Full Text 3.0 (W3C)

- Υποστηρίζει τα περισσότερα EXPath/EXQuery modules και πακέτα συστήματος
- Έχει αρχιτεκτονική τύπου Client-Server με δυνατότητες καταγραφής και διαχείρισης συναλλαγών
- APIs: RESTful API, WebDAV, XML:DB, XQJ;[5] Java, C#, Perl, PHP, Python και άλλα
- Υποστηρίζει δεδομένα τύπου: XML, HTML, JSON, CSV, Text, binary data
- Το GUI της περιλαμβάνει πολλαπλά είδη οπτικοποίησης: Treemap, table view, tree view, scatter plot

Ο BaseX server τρέχει σε Java 1.6 και είναι συμβατός με Windows, Mac OS X, Linux, OpenBSD, Debian και Ubuntu.

2.4.1. Database Layout και Αρχιτεκτονική

Η BaseX χρησιμοποιεί αναπαράσταση πίνακα για τις XML δενδρικές δομές (tree structure) για την αποθήκευση των XML εγγράφων. Η βάση λειτουργεί ως ένα container για ένα σκέτο έγγραφο ή μια συλλογή αυτών. Επιπλέον, παρέχει διάφορους τύπους δεικτών (indexes) για την βελτίωση των επιδόσεων των αναζητήσεων, των συγκρίσεων κειμένου κλπ.

Βασίζεται σε μια ισχυρή αρχιτεκτονική τύπου Client/Server και χειρίζεται με αποδοτικό τρόπο την ταυτόχρονη εγγραφή και ανάγνωση από πολλαπλούς χρήστες [41]. Ο Server είναι υψηλής απόδοσης και ταυτόχρονα χαμηλής συντήρησης. Δεν βασίζεται σε άλλες βιβλιοθήκες και έχει μικρό ίχνος μνήμης, κάτι που την καθιστά ιδανική λύση για ενσωματωμένα συστήματα και light-weight web solutions [42]. Ο Server προσφέρει κεντρική αποθήκευση για τα XML αρχεία και τα binary files. Ο χρήστης μπορεί να έχει πρόσβαση σε αυτόν μέσω clients πολλών προγραμματιστικών γλωσσών. Σε ένα περιβάλλον πολλών χρηστών, υπάρχει ένας lock manager ο οποίος υποστηρίζει ταυτόχρονες αναγνώσεις και ACID-safe write transactions. Όλες οι κινήσεις εγγραφής και ανάγνωσης καταχωρούνται με χρονολογική σειρά από τον Server. Γενικές και τοπικές άδειες εκχωρούνται στους χρήστες που συνδέονται στον Server, χρησιμοποιώντας ασφαλή έλεγχο ταυτότητας (securea uthentication). Η BaseX παρέχει ένα RESTful interface για την πρόσβαση στα δεδομένα σε τοπικές ή απομακρυσμένες XML βάσεις. Τα αποτελέσματα των επερωτήσεων εξάγονται σε μορφή JSON. Για ad-hoc πρόσβαση στις βάσεις, η BaseX προσφέρει μια WebDAV υπηρεσία, η οποία επιτρέπει στους χρήστες να αποθηκεύουν και να οργανώνουν γρήγορα τους πόρους μέσω ενός απλού WebDAV-enabled file manager.

Προσφέρει 2 ειδών interfaces για δικτυακή επικοινωνία: ένα Java Client API που χρησιμοποιεί απευθείας sockets και ένα REST interface μέσω HTTP.

Δεν δίνει μεγάλη βαρύτητα σε DTDs και XMLschemas (είναι schemaless), αρκεί το XML έγγραφο να είναι καλά δομημένο. Είναι κάτι αντίστοιχο με τη Mongo, αλλά για XML.

Η BaseX είναι μια XML βάση δεδομένων γραμμένη σε Java. Αποθηκεύει XML έγγραφα χρησιμοποιώντας ένα δικό της format στο οποίο οι πληροφορίες σχετικά με το node tree του εγγράφου αποθηκεύονται σε ένα σετ από πίνακες. Τα έγγραφα αποθηκεύονται σε μία ¨βάση¨ η οποία μπορεί να περιέχει ένα ή και περοσσότερα έγγραφα. Η βάση αυτή παρέχει το πλαίσιο για την αξιολόγηση επερωτήσεων, όπως το που μπορεί να βρίσκονται κάποια έγγραφα που ορίζονται από κάποιες συναρτήσεις Xquery (fn:doc και fn:collection) [43].

Η BaseX υποστηρίζει την XQuery, το XQuery Update Facility, και το XQuery Full Text. Οι επεκτάσεις για την XQuery περιλαμβάνουν την εκτέλεση δυναμικά κατασκευασμένων εκφράσεων XQuery, εκφράσεις try/catch και άμεσες κλήσεις εντολών Java. Οι επεκτάσεις για το XQuery Full Text περιλαμβάνουν ασαφή επερωτήματα (fuzzy queries) τα οποία εκτελούν ματσαρίσματα κειμένου κατά προσέγγιση [43].

Υποστηρίζει τέσσερα είδη καταλόγων (indexes). Τα ευρετήρια κειμένου και χαρακτηριστικού (text και attribute) κατατάσουν PCDATA και τιμές χαρακτηριστικών και χρησιμοποιούνται για την επίλυση συγκρίσεων. Και οι δύο υποστηρίζουν ακριβές αντιστοιχίες και περιοχές (ranges). Τα ευρετήρια πλήρους κειμένου χρησιμοποιούνται κατά τη διάρκεια αναζητήσεων πλήρους κειμένου και μπορούν να διαμορφωθούν για οποιοδήποτε τύπο αναζήτησης. Τα ευρετήρια διαδρομής χρησιμοποιούνται για την επίλυση αναζητήσεων τοποθεσίας και διαδρομής. Οι εφαρμογές πρέπει να ενημερώνουν ρητά τους καταλόγους/ευρετήρια μετά από κάποια διαδικασία ενημέρωσης, ώστε να εξισορροπείται η ανάγκη για ανημερώσεις με την ανάγκη για πρόσβαση [43].

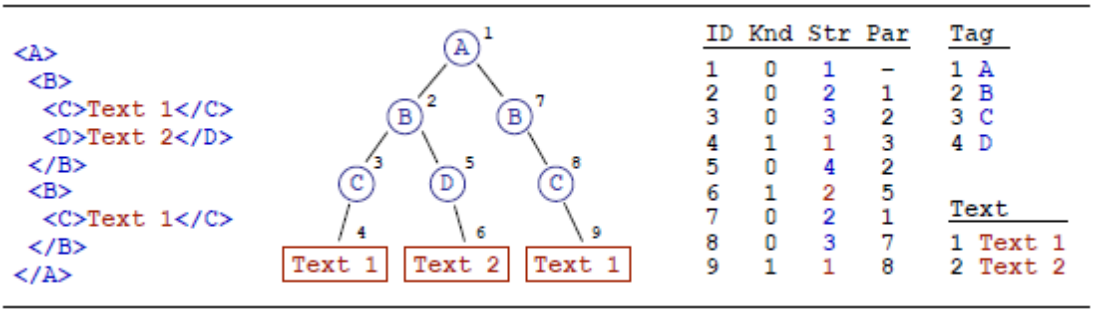
Μπορεί να λειτουργήσει ως αυτόνομη εφαρμογή επερωτήσεων ή σε λειτουργία client/server. Ως διακομιστής (server) υποστηρίζει συναλλαγές, διαχείριση χρηστών και καταγραφή (logging). Μπορεί να εκτελεστεί από την γραμμή εντολών, μέσα από ένα γραφικό εργαλείο GUI ή με την κλήση ενός API. Υποστηρίζονται τρία διαφορετικά API: το XQJ, το XML:DB και ένα δικό της API που δέχεται προγραμματιστικά εντολές τύπου command-line[43].

2.4.2. Πλεονεκτήματα και Features της BaseX

Ο χειρισμός των XMLδεδομένων τις περισσότερες φορές είναι αρκετά δύσκολος, λόγω του τεράστιου όγκου τους. Για να εξερευνήσει κανείς μια XML βάση δεδομένων χρειάζεται εργαλεία που θα κάνουν το περιεχόμενο εύκολο να διερευνηθεί, απλό να επεξεργαστεί και

θα το καταστήσουν ικανό να απεικονιστεί αποτελεσματικά. Τα εργαλεία αυτά τα προσφέρει στο σύνολό τους η BaseX.

Η BaseX παρέχει ουσιαστικά μια δενδρική (βασισμένη σε πίνακες) κωδικοποίηση XML κόμβων. Έχει δωθεί ιδιαίτερο βάρος στην αναπαράσταση του αρχικού εγγράφου να εξοικονομεί μνήμη, ώστε η επιλεγμένη κωδικοποίηση να αποτελείται μόνο από τα βασικά χαρακτηριστικά του κόμβου που είναι αναγκαία για να καταστεί δυνατή η πλήρης και γρήγορη διέλευση όλων των XPath αξόνων. Η αναπαράσταση «πίνακα» αυτή, αγνοεί πλήρως το σχήμα – δεν απαιτείται DTD ή XML Schema για την κωδικοποίηση του εγγραφου [44].



Εικόνα15: Mapping ενός XML document. αριστερά: αυθεντικό XML document, κέντρο: δενδρική απεικόνιση, δεξιά: encoding πίνακα [Gruen et al. [2007], Visually Exploring and Querying XML with BaseX]

Μια από τις περιοχές στις οποίες είναι πολύ χρήσιμη σαν λύση είναι η περιοχή των Open Data. Υπάρχει μεγάλος όγκος από δημόσια διαθέσιμα δεδομένα που βρίσκονται «παγιδευμένα» σε στατικά έγγραφα XML. Κάποια από τα πλεόνεκτήματα της BaseX αναφέρονται παρακάτω:

1. Παρέχει αποθήκευση δεδομένων υψηλής απόδοσης με ευρετήρια κειμένου, χαρακτηριστικού (attribute), πλήρους κειμένου και διαδρομής [41].

2.Αποτελεσματική υποστήριξη των W3C XPath/XQuery Recommendations, Full Text και Update επεκτάσεων [41].

3. Διαθέτει ένα από τα υψηλότερα ποσοστά «συμμόρφωσης» για όλες τις υποστηριζόμενες προδιαγραφές [41].

4. Η αρχιτεκτονική τύπου Client/Server υποστηρίζει ασφαλής ACID συναλλαγές, διαχείριση χρηστών και καταγραφή [41].

5. Παρέχει διαδραστικές απεικονίσεις και υποστηρίζει πολύ μεγάλα XML έγγραφα [41].

6. Είναι ο μόνος διαθέσιμος XQuery editor πραγματικού χρόνου, με επισήμανση σύνταξης και feedback σφαλμάτων [41].

7. Υποστηρίζει ένα μεγάλο εύρος από interfaces:REST/RESTXQ, WebDAV, XQJ, XML:DB; Και clients σε διάφορες γλώσσες [41].

Διαχείριση συναλλαγών (Transaction Management)→ Η αρχιτεκτονική τύπου client/server της BaseX προσφέρει συναλλαγές ACID safe με πολλαπλά αιτήματα ανάγνωσης και εγγραφής [45]. Μια συναλλαγή είναι ίση με μια εντολή ή μια επερώτηση. Έτσι κάθε εντολή/επερώτηση που στέλνεται στον server γίνεται αυτόματα μια συναλλαγή. Τα εισερχόμενα ερωτήματα αναλύονται και ελέγχονται για σφάλματα στον διακομιστή. Αν η εντολή ή η επερώτηση δεν είναι σωστή, το αίτημα δεν θα εκτελεστεί, και ο χρήστης θα λάβει μήνυμα σφάλματος. Διαφορετικά η αίτηση γίνεται συναλλαγή και εισέρχεται στο transaction monitor [45]. Πολλές λειτουργίες ενημέρωσης ξεκινούν από XQuery Update εκφράσεις. Κατά την εκτέλεση μιας εντολής ενημέρωσης, όλες οι επιμέρους εργασίες ενημέρωσης της εντολής αυτής, αποθηκεύονται σε μια λίστα ενημερώσεων (update list) που βρίσκεται σε εκκρεμότητα. Θα εκτελεστούν όλες με τη μια, έτσι ώστε η βάση να ενημερώνεται ατομικά. Εάν κάποια από τις υπό-εργασίες ενημέρωσης είναι λάθος, η συνολική συναλλαγή θα ματαιωθεί.

ConcurrencyControl (Συγχρονισμός)→ Η BaseX παρέχει «κλείδωμα» (locking) σε επίπεδο βάσης δεδομένων. Οι συναλλαγές εγγραφής δεν αποκλείουν απαραίτητα όλες τις άλλες συναλλαγές. Ο αριθμός των παράλληλων συναλλαγών μπορεί να περιοριστεί με την επιλογή της παράλληλης ρύθμισης (PARALLEL) [45].

Transaction Monitor→ Το Monitor συναλλαγής διασφαλίζει ότι μόνο μια συναλλαγή εγγραφής ή μια αυθαίρετη ποσότητα συναλλαγών ανάγνωσης ανά βάση δεδομένων είναι ενεργές την ίδια χρονική στιγμή. Τα αδιέξοδα προλαμβάνονται χρησιμοποιώντας δύο φάσης κλειδώματος (two phase locking). Λόγω των ιδιαίτερων χαρακτηριστικών του XQuery Update, όλες οι ενημερώσεις συντάσσονται στο τέλος της επερώτησης. Τα locks δεν είναι συγχρονισμένα μεταξύ πολλών instances της BaseX. Η αρχιτεκτονική client/server είναι προτιμότερη όταν πρέπει να εκτελεστούν εντολές ταυτόχρονης εγγραφής [45].

XQuery Processor→ Ένα από τα βασικά χαρακτηριστικά και δυνατά σημεία της BaseX είναι το γρήγορο και αποτελεσματικό XQuery Processing. Η εστίαση του στα indexes επιτρέπει ένα ευρύ φάσμα τεχνικών βελτιστοποίησης στα αιτήματα XQuery του χρήστη. Τι υποστηρίζει [46]:

XQuery 3.1: Νέα χαρακτηριστικά γλώσσας, συμπεριλαμβανομένων της ομαδοποίησης (groupby), try/catch, switch, function items, χάρτες, πίνακες, υποστήριξη για JSON.

FullText: εφαρμογή του W3C XQuery Full Text. Η επέκταση full-text της XQuery κάνει την BaseX ιδανικό εργαλείο για την ανάπτυξη υψηλής απόδοσης συστημάτων ανάκτησης πληροφοριών

Updates: πολύ αποτελεσματική εφαρμογή του XQuery Update Facility για την εισαγωγή, διαγραφή, μετονομασία και την αντικατάσταση των κόμβων

Java Bindings: η άμεση πρόσβαση στις Java διαδικασίες και ο χειρισμό αντικειμένων παρέχει σχεδόν ατελείωτη επεκτασιμότητα και απλή ενσωμάτωση εξωτερικών βιβλιοθηκών της Java.

Packaging: Υποστήριξη για το τυποποιημένο EXPath Packaging Module που επιτρέπει την απλή και εύκολη επέκταση του XQuery processor.

Errors: το λεπτομερές και ολοκληρωμένο feedback λαθών επιταχύνει την ανάπτυξη και τη βελτίωση των XQuery / XML εφαρμογών. Σε αυτό βοηθά η πραγματικού χρόνου αξιολόγηση συντακτικού στον query editor

Indexes: τα πολύ συμπαγή ευρετήρια κειμένου, χαρακτηριστικού, πλήρους κειμένου και διαδρομής επιταχύνουν πάρα πολύ τη διαδικασία αξιολόγησης. Υπάρχει ακόμα και η επιλογή fuzzy-match του πλήρους κειμένου που επιτρέπει κατά προσέγγιση αναζητήσεις και ανάκτηση.

Execution Plan: Κάθε διαδικασία αξιολόγησης μπορεί να απεικονιστεί με επιμέρους βήματα και με index accesses, τα οποία προσφέρουν περαιτέρω βελτιστοποίηση. Τα σχέδια εκτέλεσης μπορούν να έχουν μορφή απλού κειμένου, XML και .dot μορφή.

HTTP: με ενσωματωμένη υποστήριξη για το HTTP, οι πηγές πληροφόρησης σε όλο το web είναι προσβάσιμες μέσω της XQuery.

JSON: η επέκταση JSON προσφέρει υποστήριξει στην μετατροπή XML σε JSON και αντίστροφα.

Cryptography: η επέκταση αυτή παρέχει λειτουργίες XQuery για την ψηφιακή υπογραφή εγγράφων XML. Επιπλέον, επιτρέπει στους χρήστες να κρυπτογραφήσουν και να αποκρυπτογραφήσουν μηνύματα και βασίζεται σε διάφορους αλγόριθμους κρυπτογράφησης.

File: η ενότητα αυτή περιέχει λειτουργίες XQuery και μεταβλητές που σχετίζονται με διάφορες λειτουργίες του συστήματος αρχείων, όπως η ανάγνωση ή εγγραφή αρχείων και καταλόγων.

Database: περίπλοκες λειτουργίες της βάσης δεδομένων μπορούν να διαμορφωθούν και να επεξεργαστούν μέσω της XQuery.

Map: η ενότητα αυτή προσφέρει μια αποτελεσματική και λειτουργική εφαρμογή των χαρτών στην XQuery, μια δομή δεδομένων χρησιμοποιείται σε μεγάλο βαθμό στις περισσότερες γλώσσες προγραμματισμού.

Higher Order Functions: η BaseX εφαρμόζει πλήρως τα XQuery 3.0 function items, τα οποία μπορούν να χρησιμοποιηθούν για να γραφούν προσαρμοσμένοι τύποι δεδομένων

SQL: η επέκταση της SQL περιέχει λειτουργίες XQuery για την πρόσβαση σε σχεσιακές βάσεις δεδομένων μέσω Xquery, με τη χρήση SQL. Επιτρέπει στους προγραμματιστές να εκτελούν εντολές SQL και να ανακτούν τα αποτελέσματα για περαιτέρω επεξεργασία.

XSLT: η BaseX δίνει στους προγραμματιστές πλήρη πρόσβαση για να μετατρέπουν XML με XSLT 1.0 και XSLT 2.0 (μέσω Saxon).

BaseX Graphical User Interface (GUI)→ η BaseX μπορεί να χρησιμοποιηθεί από τη γραμμή εντολών, σε χρήση client / server, ή χρησιμοποιώντας την γραφική διεπαφή χρήστη (GUI). Εκτός από την κύρια λειτουργικότητα της βάσης δεδομένων, το γραφικό περιβάλλον προσφέρει καινοτόμο γραφικό frontend για να απεικονίσει τα δεδομένα XML. Ένα προηγμένο ενσωματωμένο XQuery επεξεργαστή συνδέεται με τις απεικονίσεις, παρέχοντας άμεση feedback με τα αποτελέσματα [47].

Highly Interactive Visualization: Εξερευνεί, αναλύει και επιτρέπει την περιήγηση στα XML δεδομένα

Real time Execution: οι Xquery εκφράσεις μπορούν να εκτελεστούν κατά την πληκτρολόγηση τους, και τα αποτελέσματα είναι άμεσα ορατά.

XQuery, XML and Script Editor: ο ενσωματωμένος επεξεργαστής παρέχει επισήμανση σύνταξης και διαθέτει λεπτομερές feedback για τον εντοπισμό σφαλμάτων στα επερωτήματα.

Keyword Search: γρήγορα ad-hoc αιτήματα: απλή εισαγωγή λέξεων κλειδιά για αναζήτηση μέσα στη δομή και το περιεχόμενο των XML εγγράφων.

XPath Suggestions: προσφέρει αυτόματα το επόμενο πιθανό βήμα κατά την διάρκεια γραφής της επερώτησης.

Server Administration: ο χρήστης μπορεί να διαχειρίζεται το διακομιστή της βάσης δεδομένων από μια γραφική διεπαφή χρήστη.

Internationalization: υποστηρίζει πολλές γλώσσες (Αγγλικά, Γερμανικά, Γαλλικά, Ιταλικά, Ολλανδικά, Ιαπωνικά, Ινδονησιακά, Ρουμανικά, Μογγολικά)

Ένα από τα κύρια μειονεκτήματα της BaseX είναι ότι δεν προσφέρει καμία λύση για κατανεμημένα περιβάλλοντα. Έτσι ο χρήστης θα πρέπει να παρέχει μόνος του στο σύστημα κάποιου είδους βασικού sharding strategy.

2.4.3. Σύγκριση XML και Σχεσιακών Βάσεων

Και τα δύο είδη βάσεων χρησιμοποιούν καθιερωμένες στον χώρο τεχνικές για να εξάγουν τα δεδομένα που περιέχουν.

Οι σχεσιακές βάσεις είναι καλύτερες για τον χειρισμό μεγάλων όγκων δεδομένων μέσα σε ένα σύστημα. Διαθέτουν ώριμα συστήματα διαχείρισης που μπορούν με αποτελεσματικό και αξιόπιστο τρόπου να διατηρήσουν μεγάλες ποσότητες δομημένων δεδομένων. Αυτά τα δεδομένα μπορούν να ενημερωθούν μέσω συναλλαγών που διασφαλίζουν την ακεραιότητα της βάσης και το περιεχόμενο μπορεί να εξαχθεί πολύ γρήγορα με τις κατάλληλες ρυθμίσεις. Δεν υπάρχει ισοδύναμο σύστημα διαχείρισης XML, και η άμεση χρήση XML εγγράφων για την αποθήκευση και διατήρηση μεγάλου όγκου δεδομένων μπορεί να αποδειχθεί αναποτελεσματική και αναξιόπιστη [48].

Τα XML έγγραφα περιέχουν τόσο τα στοιχεία όσο και την σχέση αυτών των δεδομένων με τέτοιο τρόπο ώστε και οι μηχανές και οι άνθρωποι να μπορούν να διαβάσουν. Ένα έγγραφο XML μπορεί να διαβιβαστεί ηλεκτρονικά και όλες οι πληροφορίες μεταφέρονται με αυτό (είναι δηλαδή self describing). Με βάση το χαρακτηριστικό, η XML έχει γίνει ένα βασικό πρότυπο για τις αρχιτεκτονικές Service Oriented (SOA) και τα Web Services [48].

Οι σχεσιακές βάσεις χρησιμοποιούνται μέσα σε ένα σύστημα για την διαχείρηση και την διατήρηση μιας μεγάλης συλογής δεδομένων, ή για την διαχείρηση μιας βάσης στην οποία γίνονται συχνές ενημερώσεις από πολλαπλούς χρήστες. Οι XML βάσεις χρησιμοποιούνται συνήθως μεταξύ των components κατανεμημένων συστημάτων

Το μεγαλύτερο μέρος της επιρροής της XML στον λογικό και φυσικό σχεδιασμό μιας βάσης δεδομένων βασίζεται σε θεμελιώδεις ιδιότητες της XML που το κάνουν να διαφέρει από το σχεσιακό μοντέλο[49]:

Τα XML έγγραφα είναι self describing: Ένα συγκεκριμένο έγγραφο περιέχει όχι μόνο τα δεδομένα, αλλά και όλα τα απαραίτητα μεταδεδομένα. Ως αποτέλεσμα, ένα XML έγγραφο μπορεί να αναζητηθεί ή να ενημερωθεί χωρίς να απαιτείται στατικός ορισμός σχήματος. Τα σχεσιακά μοντέλα, από την άλλη πλευρά, απαιτούν στατικούς ορισμούς σχήματος. Όλες οι σειρές του πίνακα πρέπει να έχουν το ίδιο σχήμα.

Η XML είναι ιεραρχική γλώσσα: Ένα δεδομένο έγγραφο δεν αντιπροσωπεύει μόνο βασικές πληροφορίες, αλλά επίσης πληροφορίες σχετικά με τη σχέση των στοιχείων των δεδομένων, με τη μορφή ιεραρχίας. Τα σχεσιακά μοντέλα απαιτούν όλες οι πληροφορίες των σχέσεων να εκφράζονται είτε μέσω σχέσεων πρωτεύοντος ή ξένου κλειδιού, είτε εκπροσωπώντας αυτές τις πληροφορίες σε άλλες σχέσεις.

Στην πραγματικότητα δεν υπερτερεί κάποιο μοντέλο σε σχέση με το άλλο. Τα XML και τα σχεσιακά μοντέλα μπορούν να θεωρηθούν συμπληρωματικές λύσεις. Ορισμένα δεδομένα έχουν εγγενώς ιεραρχική φύση ενώ άλλα δεδομένα έχουν εγγενώς σχέση πίνακα. Κάποια δεδομένα έχουν πιο άκαμπτο σχήμα, ενώ άλλα έχουν λιγότερο αυστηρό σχήμα. Ορισμένα στοιχεία πρέπει να ακολουθούν μια συγκεκριμένη σειρά, ενώ άλλα όχι [49].

Υπάρχουν περιπτώσεις όπου η χρήση XML βάσεων δεδομένων συνίσταται. Οι περιπτώσεις αυτές περιλαμβάνουν τα εξής σενάρια:

- Όταν το σχήμα είναι ασταθές → οι αλλαγές στο σχήμα στο σχεσιακό μοντέλο πολλές φορές είναι δύσκολες και προυποθέτουν downtime της βάσης.
- Όταν τα δεδομένα έχουν εγγενώς ιεραρχική φύση
- Όταν οι εφαρμογές έχουν αραιά χαρακτηριστικά → κάποιες εφαρμογές έχουν μεγάλο αριθμό από πιθανά attributes, από τα οποία πολλά μπορεί να είναι κενά για κάποιες τιμές.
- Όταν δεδομένα μικρού όγκου είναι αυστηρά δομημένα

Πολλοί συμβιβασμοί προκύπτουν από τις θεμελιώδης διαφορές μεταξύ του XML και του σχεσιακού μοντέλου.Ο κύριος συμβιβασμός είναι η ευελιξία σε σχέση με την απόδοση. Η XML ως αυτο-περιγραφούμενη δομή δεδομένων, επιτρέπει διαφορετικές μορφές των δεδομένων να αποθηκεύονται σε ένα ενιαίο έγγραφο ή σειρά, χωρίς να θυσιάζεται η

δυνατότητα αναζήτησης σε τμήματα των εν λόγω δεδομένων. Ωστόσο, αυτή η ευελιξία οδηγεί σε μεγαλύτερη χρόνο λειτουργίας της CPU και αυξάνει το κόστος I / O για να ερμηνεύτουν τα δεδομένα. Το σχεσιακό μοντέλο, δεδομένου του πιο άκαμπτου σχηματός του, απαιτεί σημαντικά λιγότερη ερμηνεία και για το λόγο αυτό αποδίδει καλύτερα. Ωστόσο, σε ορισμένες περιπτώσεις, η XML βελτιώνει την επίδοση συγκριτικά με το σχεσιακό μοντέλο ακριβώς λόγω της ευελιξίας της. Τα σχεσιακά μοντέλα συχνά απαιτούν κανονικοποίηση των δεδομένων για να τα προσαρμόσουν καλύτερα στο μοντέλο. Κάθε φορά που τα δεδομένα κανονικοποιούνται, υπάρχει μεγάλη πιθανότητα να χρειάζονται joins για να συνδεθούν τα δεδομένα, και τα joins κοστίζουν περισσότερο [49].

3. Μελέτη Περίπτωσης

3.1. Ποιοτική Σύγκριση

Στο κεφάλαιο αυτό θα κάνουμε μια ποιοτική σύγκριση των NoSQL βάσεων δεδομένων Cassandra και BaseX, αξιολογώντας αυτές με βάση 6 μεγάλες κατηγορίες χαρακτηριστικών:

System→ Server Operating System, Programming language

Design & Features→ Schema, Query language, Protocol, MapReduce, REST, Other API and access methods , Other features, Data Processing nature

Integrity (ACID properties)→ Integrity model, Atomicity, Isolation, Durability (data storage), Transactions, Referential integrity, Revision Control, Single point of failure, Versioning, Concurrency Control

Indexing→ Indexing, Secondary indexes, Composite keys, Full text search, Geospatial indexes, Graph Support

Distribution→ Horizontal scalable, Replication, Replication mode, Sharding, Shared nothing architecture, Partitioning, CAP characteristics

Security Features→Encryption, Authentication, Authorization, Auditing

Κάποια χαρακτηριστικά είναι ιδιαιτέρου ενδιαφέροντος για την επιλογή μιας βάσης ανάλογα με τις απαιτήσεις της κάθε εφαρμογής και για το λόγο αυτό θα προχωρήσουμε σε μια σύντομη περιγραφή του καθενός.

Querying: η ικανότητα επερωτήσεων ενός συστήματος βάσης δεδομένων διαδραματίζει σημαντικό ρόλο όταν ο χρήστης πρέπει να επιλέξει μεταξύ των διαφόρων ειδών για ένα

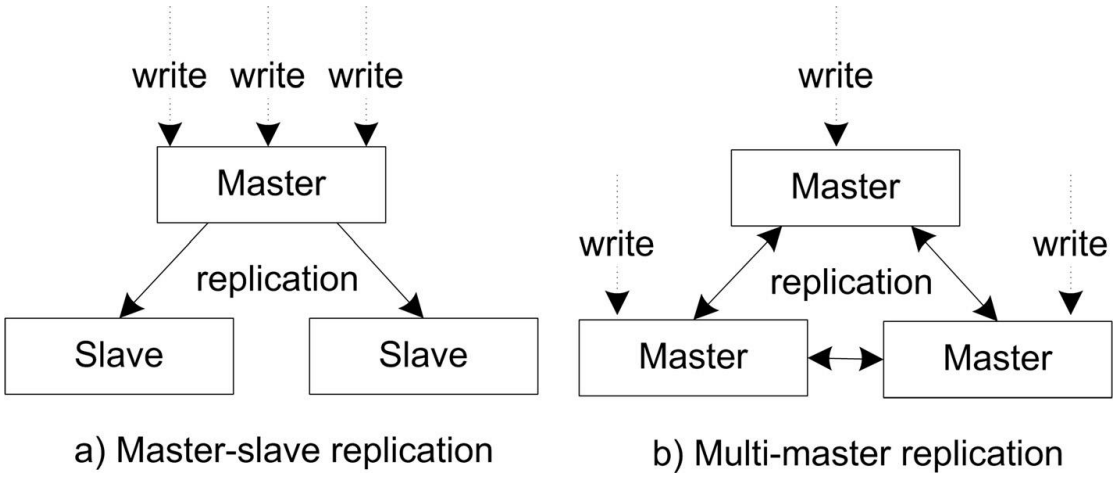
συγκεκριμένο σενάριο χρήσης. Κάθε διαφορετική βάση προσφέρει διαφορετικά APIs και διεπαφές για να αλληλεπιδρά με τον χρήστη. Η ικανότητα αυτή εξαρτάται άμεσα από το μοντέλο δεδομένων στο οποίο στηρίζεται σχεδιαστικά μια συγκεκριμένη βάση [50]. Ένα σημαντικό επίσης χαρακτηριστικό είναι το επίπεδο υποστήριξης του MapReduce, το οποίο θεωρείται η πλέον αποδεκτή προσέγγιση για την επεξεργασία κατανεμημένων δεδομένων σε ένα σύμπλεγμα από ηλεκτρονικούς υπολογιστές [51]. Ομοίως, τα SQL-like επερωτήματα θεωρούνται μια προτιμώμενη επιλογή, λόγω της διαδεδομένης τους χρήσης τις τελευταίες δεκαετίες μέσα από το σχεσιακό μοντέλο βάσεων δεδομένων.

Scaling: Ένα από τα κύρια χαρακτηριστικά των NoSQL συστημάτων είναι η ικανότητά της οριζόντιας και αποτελεσματικής κλιμάκωσης με την προσθήκη περισσότερων servers. Όσον αφορά την κλιμάκωση, μπορούμε να θεωρήσουμε τρεις διαστάσεις κλιμάκωσης: κλιμάκωση αιτημάτων ανάγνωσης, κλιμάκωση αιτημάτων εγγραφής, ή κλιμάκωση αποθήκευσης δεδομένων. Οι στρατηγικές partitioning, replication, consistency, and concurrency control που χρησιμοποιούνται από τα NoSQL συστήματα έχουν σημαντικό αντίκτυπο στην επεκτασιμότητα τους. Για παράδειγμα, το partitioning καθορίζει την κατανομή των δεδομένων σε πολλούς διακομιστές και είναι ως εκ τούτου ένα μέσο για την επίτευξη και των τριών διαστάσεων κλιμάκωσης [50]. Ένας άλλος σημαντικός παράγοντας για την κλιμάκωση των αιτημάτων ανάγνωσης και εγγραφής είναι το Replication, δηλαδή η αποθήκευση ίδιων δεδομένων σε πολλαπλούς διακομιστές έτσι ώστε τα αιτήματα ανάγνωσης και εγγραφής να διανέμονται σε αυτούς. Η αντιγραφή (Replication) παίζει επίσης σημαντικό ρόλο στην ανοχή σφαλμάτων, διότι η διαθεσιμότητα των δεδομένων μπορεί να αντέξει την αποτυχία ενός ή και περισσότερων διακομιστών. Επιπλέον, η επιλογή του μοντέλου αντιγραφής είναι έντονα συνδεδεμένη με το επίπεδο συνέπειας που παρέχεται από την βάση. Τέλος, ένας άλλος σημαντικός παράγοντας στην κλιμάκωση είναι το concurrency control. Οι απλές τεχνικές τεχνικές κλειδώματος (lock) ανάγνωσης / εγγραφής μπορεί να μην παρέχουν επαρκή concurrency control για υψηλές αποδόσεις. Ως εκ τούτου, πολλές βάσεις χρησιμοποιούν πιο προηγμένες τεχνικές, όπως optimistic locking with multi-version concurrency control (MVCC) [50].

Partitioning: Οι περισσότερες NoSQL βάσεις δεδομένων υλοποιούν κάποιο είδος οριζόντιας τμηματοποίησης (horizontal partitioning ή sharding), η οποία περιλαμβάνει την αποθήκευση από σετ ή σειρές / εγγραφές σε διαφορετικά τμήματα (ή shards) που μπορεί να βρίσκονται σε διαφορετικούς εξυπηρετητές. Οι δύο πιο κοινές στρατηγικές οριζόντιας τμηματοποίησης είναι το range partitioning (διαμερισμός εύρους) και το consistent hashing

(συνεπής κατακερματισμός). Το range partitioning αποδίδει δεδομένα στις κατατμήσεις που κατοικούν σε διάφορους servers και βασίζεται στο εύρος ενός κλειδιού διαμέρισης. Ο server είναι υπεύθυνος για την αποθήκευση και για τον χειρισμό των αιτημάτων ανάγνωσης/εγγραφής ενός συγκεκριμένου εύρους κλειδιών. Στο consistent hashing, το σύνολο δεδομένων αναπαρίσταται ως ένας κύκλος ή δακτύλιος. Ο δακτύλιος αυτός χωρίζεται σε έναν αριθμό τμημάτων ίσο με τον αριθμό των διαθέσιμων κόμβων, και κάθε κόμβος αντιστοιχίζεται σε ένα σημείο πάνω στο δακτύλιο. Με το consistent hashing, η θέση ενός αντικειμένου μπορεί να υπολογίζεται πολύ γρήγορα, και δεν υπάρχει ανάγκη για μια χαρτογράφηση όπως στο range partitioning [50].

Replication: Εκτός από την αύξηση της κλιμάκωσης στην ανάγνωση/εγγραφή, το replication βελτιώνει επίσης την αξιοπιστία του συστήματος, την ανοχή σφαλμάτων, και την αντοχή. Δύο βασικές προσεγγίσεις μπορούν να διακριθούν για το replication: master-slave και multi-master replication. Στο master-slave replication, ένας κόμβος έχει οριστεί ως κύριος (master) και είναι ο μοναδικός που επεξεργάζεται τα αιτήματα εγγραφής. Οι όποιες αλλαγές περνάνε από τον κύριο κόμβο στους βοηθητικούς (slaves). Στο multi-master replication, πολλαπλοί κόμβοι μπορούν να επεξεργαστούν τα αιτήματα εγγραφής, τα οποία περνάνε και στους υπόλοιπους κόμβους [50].



Εικόνα 16: Replication modes [Grolinger et al. Journal of Cloud Computing: Advances, Systems and Applications 2013, 2:22]

Consistency: Η συνέπεια, ως μία από τις ACID ιδιότητες, εξασφαλίζει ότι μια συναλλαγή φέρνει τη βάση δεδομένων από μια έγκυρη κατάσταση σε μια άλλη. Η συνέπεια όπως χρησιμοποιείται στο θεώρημα CAP, σχετίζεται με τον τρόπο που τα δεδομένα εμφανίζονται μεταξύ των κόμβων του διακομιστή μετά από πράξεις ενημέρωσης. Διακρίνονται δύο μοντέλα συνοχής/συνέπειας: η ισχυρή και η ενδεχόμενη συνέπεια. Η ισχυρή ή άμεση

συνέπεια (υπάρχει στο Synchronous replication [52]) εξασφαλίζει ότι όταν επιβεβαιώνονται αιτήματα εγγραφής, τα ίδια (ενημερωμένα) δεδομένα είναι ορατά σε όλα τα αιτήματα ανάγνωσης που ακολουθούν. Στο μοντέλο της ενδεχόμενης συνέπειας (εμφανίζεται στο asynchronous replication[52]), οι αλλαγές διαδίδονται μέσω του συστήματος σε επαρκή χρόνο. Ως εκ τούτου, κάποιοι κόμβοι του server μπορεί να περιέχουν ασυνεπή (μη ενημερωμένα) δεδομένα για κάποιο χρονικό διάστημα [50].

Concurrency control: Το Concurrency control έχει ιδιαίτερο ενδιαφέρον για τα NoSQL συστήματα, επειδή αυτά συνήθως πρέπει να εξυπηρετούν ένα μεγάλο αριθμό ταυτόχρονων χρηστών και να έχουν υψηλά ποσοστά σε ανάγνωση και εγγραφή. Όλα τα NoSQL συστήματα διευκολύνουν τον συγχρονισμό με την εφαρμογή partitioning και replication. Τα βασικά μοντέλα ελέγχου συγχρονισμού μπορούν να κατηγοριοποιηθούν ως pessimistic ή optimistic. Ο pessimistic έλεγχος συγχρονισμού, ή αλλιώς το pessimistic locking, υποθέτει ότι δύο ή περισσότεροι ταυτόχρονοι χρήστες θα προσπαθήσουν να ενημερώσουν το ίδιο αρχείο την ίδια χρονική στιγμή. Για να αποφευχθεί αυτή η κατάσταση, ένα lock τοποθετείται πάνω στο προσπελάσιμο αρχείο ώστε να είναι εγγυημένη η αποκλειστική πρόσβαση. Ο optimistic έλεγχος συγχρονισμού ή optimistic locking υποθέτει ότι οι συγκρούσεις είναι πιθανές, αλλά σπάνιες. Ως εκ τούτου, αντί να κλειδώνει το αρχείο, η βάση κάνει έλεγχο στο τέλος της λειτουργίας για να διαπιστωθεί αν ταυτόχρονοι χρήστες έχουν προσπαθήσει να τροποποιήσουν το ίδιο αρχείο. Σε περίπτωση σύγκρουσης, διαφορετικές στρατηγικές επίλυσης συγκρούσεων μπορούν να χρησιμοποιηθούν, όπως για παράδειγμα να αποτύχει η λειτουργία αμέσως ή ξαναδοκιμαστεί μία από τις πράξεις [50]. Υπάρχουν τα εξής ενδεχόμενα [52]:

- Locks→ δεν επιτρέπουν σε πάνω από έναν ενεργό χρήστη να επεξεργαστεί ένα αρχείο.
- MVCC (Multi-Version Concurrency Control)→ εγγυάται μια συνεκτική εικόνα της βάσης δεδομένων στην ανάγνωση, αλλά μπορεί να οδηγήσει σε αντικρουόμενες εκδοχές μιας οντότητας, εάν πολλαπλοί χρήστες δοκιμάσουν ταυτόχρονα να την τροποποιήσουν. Αυτό σημαίνει ότι η συνοχή μιας συναλλαγής διατηρείται ακόμη και αν παρουσιάζονται ποικίλα στιγμιότυπα σε διαφορετικούς χρήστες σε κάθε δεδομένη χρονική στιγμή. Οποιεσδήποτε αλλαγές γίνονται στη βάση δεδομένων θα εμφανίζονται στους άλλους ανάλογα με το σε ποιο στιγμιότυπο αυτοί αναφέρονται.

- None→ η ατομικότητα λείπει σε μερικά συστήματα με αποτέλεσμα να μην παρέχετε η ίδια εικόνα της βάσης δεδομένων σε πολλαπλούς χρήστες όταν θέλουν να την επεξεργαστούν.
- ACID→ για αξιόπιστες συναλλαγές με τη βάση, οι ιδιότητες ACID είναι ένα ασφαλές σενάριο. Παρέχουν τη δυνατότητα για συναλλαγές με προκαταρκτικό έλεγχο για την αποφυγή συγκρούσεων χωρίς αδιέξοδα.

Security: Η ασφάλεια είναι ένα από τα βασικά χαρακτηριστικά ενός συστήματος βάσεων δεδομένων και συνήθως αγνοείται από τις περισσότερες NoSQL υλοποιήσεις. Σχετικά με την ασφάλεια ενός συστήματος 4 βασικά χαρακτηριστικά μπορούν να ξεχωρίσουν [50]:

- **Authentication**: μηχανισμοί που επιτρέπουν την επαλήθευση της ταυτότητας των χρηστών οι οποίοι έχουν πρόσβαση στα δεδομένα.
- **Authorization**: αυτό αναφέρεται στην ικανότητα του συστήματος να εξασφαλίσει έλεγχο πρόσβασης στους πόρους δεδομένων.
- **Encryption**: αυτό αναφέρεται σε μηχανισμούς που κρυπτογραφούν τα δεδομένα έτσι ώστε να μην μπορούν να διαβαστούν από μη εξουσιοδοτημένα πρόσωπα
- **Auditing**: οι λειτουργίες λογιστικού ελέγχου συνήθως σχετίζονται με τη δημιουργία μιας διαδρομής ελέγχου, που καταγράφει αρχεία γεγονότων, που συνέβησαν στις βάσεις δεδομένων.

Πίνακας 3: Ποιοτική σύγκριση μεταξύ Cassandra και BaseX

	Cassandra	**BaseX**
Μοντέλο Βάσης (Database model)	Wide column store (column oriented)	Native XML DBMS
Αρχική Έκδοση (Initial Release)	2008	2007
System		
Λειτουργικό Σύστημα Διακομιστή (Server Operating System)	Cross-platform [53] BSD Linux OS X Windows	Linux OS X Windows Cross-platform [54]
Γλώσσες Προγραμματισμού (Programming language)	C# C++ Clojure Erlang Go Haskell Java JavaScript Perl	Actionscript C C# Haskell Java JavaScript Lisp Perl PHP

	PHP Python Ruby Scala	Python Qt Rebol Ruby Scala Visual Basic
Γλώσσα Δημιουργίας (Written in)	Java	Java
Άδεια (License)	Open source: Apache 2.0 license	Open source: BSD licence
Μέγιστο μέγεθος τιμής (Value max size (or document size limit))	2GB [53]	140MB [54]
Design & Features		
Πως αποθηκεύονται τα δεδομένα (Data storage)	FileSystem	FileSystem
Υποστηριζόμενοι Τύποι Δεδομένων (Datatypes)	AllMySQL JSON BLOB user-defined STATIC COLUMN [54]	XML
Δυνατότητα Ενσωμάτωσης (αν μπορεί να χρησιμοποιηθεί σε standalone program)	Όχι	Ναι
Σχήμα (Schema)	Schema-free	Schema-free
Γλώσσα Επερωτήσεων (Query language)	API calls, CQL, Thrift [53]	XQuery REST API calls [54]
Πρωτόκολλο (Protocol)	Thrift & custom binary CQL3 [53]	
Ενημέρωση εγγραφών υπό προυπόθεση (Conditional entry updates)	Ναι	Ναι
MapReduce	Ναι	Όχι
Unicode	Ναι	Ναι
Συμπίεση (Compression)	Ναι	Ναι
REST	Third party APIs [50]	Ναι
Άλλα API και	CLI και APIσε διάφορες γλώσσες.	Java API

μέθοδοι πρόσβασης (Other API and access methods)	Υποστηρίζει Thrift interface [50]	RESTful HTTP API RESTXQ WebDAV XML:DB XQJ [55]
Άλλα χαρακτηριστικά (Other features)	Μηχανισμοί Secondary indexing περιλαμβάνουν column families, super-columns, collections [50]	
Φύση επεξεργασίας δεδομένων (Data Processing nature)	Streaming και atomic batches [56]	
Integrity (ACID properties)		
Μοντέλο Ακεραιότητας (Integrity model)	BASE [53]	ACID [54]
Ατομικότητα (Atomicity)	Ναι	Ναι
Συνοχή/Συνέπεια (Consistency)	Ναι Διαμορφώσιμη, ανάλογα με τις συνολικές αιτήσεις για ανάγνωση και εγγραφή [50]	Ναι
Απομόνωση (Isolation)	Ναι	Ναι
Αντοχή (Durability (data storage))	Ναι	Ναι
Συναλλαγές (Transactions)	Όχι (η ατομικότητα και η απομόνωση υποστηρίζονται για μεμονωμένες εργασίες)	Ναι (πολλαπλοί χρήστες διαβάζουν, ένας μόνο γράφει)
Αναφορική Ακεραιότητα (Referential integrity)	Όχι	Ναι
Έλεγχος αναθεώρησης (Revision Control)	Ναι	Όχι
Μοναδικό σημείο αποτυχίας (Single point of failure)	Όχι	
Δημιουργία εκδόσεων (Versioning)	Ναι (εξωτερικό timestamp σε επίπεδο επερώτησης, αλλά δεν υπάρχει ενσωματωμένη λειτουργία) [56]	
Έλεγχος συγχρονισμού (Concurrency	Τα timestamps χρησιμοποιούνται για να καθορίσουν την πιο πρόσφατη ενημέρωση σε μια	Ναι Lock on write

Control)	στήλη. Το τελευταίο timestamp κερδίζει πάντα [50]	
Indexing		
Indexing	Basic primary και secondary [56]	
Secondary indexes	Ναι	Ναι
Σύνθετα Κλειδιά (Composite keys)	Ναι	
Αναζήτηση Πλήρους κειμένου (Full text search)	Όχι	Ναι
Geospatial indexes	Όχι	
Υποστήριξη Γράφων (Graph Support)	Όχι	
Distribution		
Επεκτασιμότητα (Horizontal scalable)	Ναι (κλιμακώνεται σταδιακά και υποστηρίζει autosharding)	Όχι
Αντιγραφή (Replication)	Ναι Peer-to-Peer	Όχι
Είδος Λειτουργίας Αντιγραφής (Replication mode)	Master-Slave-Replication [53] Masterless, asynchronous replication. 2 στρατηγικές για τα αντίγραφα: τα αντίγραφα τοποθετούνται στους επόμενους R nodes του ring, ή, τα αντίγραφα τοποθετούνται στον πρώτο node του ring που ανήκει σε άλλο data center[50]	Καμία
Sharding	Ναι	Όχι
Shared nothing architecture	Ναι	Όχι
Τμηματοποίηση (Partitioning)	Consistent hashing και range partitioning (partitioning με διατήρηση της σειράς) [50]	Καμία
Χαρακτηριστικά CAP	High Availability Partition, Tolerance, Persistence	
Security Features		
Κρυπτογράφηση (Encryption)		
Data at rest	Μόνο στο Enterprise Edition. Τα commitlogs δεν είναι κρυπτογραφημένα [50]	
Client/Server	Ναι, βασισμένο σε SSL [50]	
Server/Server	Ναι, διαμορφώσιμο: επικοινωνία server to server, μόνο μεταξύ των datacenters ή μεταξύ των servers του ίδιου rack [50]	

Πιστοποίηση (Authentication)	Ναι, τα credentials βρίσκονται σε έναν πίνακα του συστήματος. Δυνατή η παροχή pluggable implementations [50]	
Εξουσιοδότηση (Authorization)	Ναι, παρόμοια με την προσέγγιση της SQL GRANT/REVOKE. Δυνατή η παροχή pluggable implementations [50]	Οι χρήστες διαθέτουν fine-grained εξουσιοδότηση σε 4 επίπεδα [55]
Έλεγχος (Auditing)	Μόνο στο Enterprise edition. Βασισμένος στο log4j framework. Οι κατηγορίες Logging περιλαμβάνουν ADMIN, ALL, AUTH, DML, DDL, DCL and QUERY. Δυνατότητα απενεργοποίησης logging για συγκεκριμένα keyspaces.[50]	

3.2. Ποσοτική Σύγκριση

3.2.1. Εγκατάσταση Βάσεων

Το μηχάνημα που χρησιμοποιήθηκε για να γίνει η εγκατάσταση των βάσεων έχει τα εξής χαρακτηριστικά:

- **hdd**: toshiba 6GB sata3
- **cpu**: AMD FX 6300(AM3+,3,5 GHz,14 Mb)
- **ram**: 8 GB gskill 1600
- **OS:** Ubuntu 12.04.2 LTS

Για τις εκδόσεις των βάσεων μπήκαν τα τελευταία versions (Cassandra 2.1.5 και BaseX 8.2 και Mongo 3.0.2) με βάση τα εκτελέσιμα που δίνει το developing team για την κάθε βάση. Εγκαταστάθηκαν επίσης όλες οι τελευταίες εκδόσεις των Java, Maven και Git ώστε όλες οι εξαρτήσεις (dependencies) να λειτουργούν σωστά.

Για τα workload tests ο client πρέπει να βρίσκεται στην ίδια γεωγραφική θέση με το cluster της βάσης ώστε να αποφεύγονται οι καθυστερήσεις μέσω Internet (Internet latencies). Στην συγκεκριμένη εγκατάσταση, ο client και ο server βρίσκονται στο ίδιο μηχάνημα.

Για την εγκατάσταση της Cassandra χρησιμοποιήθηκε η έκδοση 2.1.5. και ο Thrift client. Για τη Mongo όλα τα τεστ έγιναν με χρήση του synchronous mode και του journaling για τα replica sets.

3.2.2. Benchmarking Tool

Στην παρούσα εργασία για το κομμάτι της ποσοτικής σύγκρισης των βάσεων, χρησιμοποιήθηκε το Yahoo! Cloud Serving Benchmark (YCSB) framework. Το YCSB είναι μια ανοιχτού κώδικα (διαθέσιμο στο Github: https://github.com/brianfrankcooper/YCSB), γραμμένη σε Java, προδιαγραφή, που χρησιμοποιείται για την αξιολόγηση των δυνατοτήτων κάποιου προγράμματος. Συχνά χρησιμοποιείται για να συγκρίνει σχετικές επιδόσεις των NoSQL συστημάτων διαχείρισης βάσεων δεδομένων [57]. Το αρχικό Benchmark αναπτύχθηκε στο τμήμα έρευνας της Yahoo! και κυκλοφόρησε το 2010 με στόχο «να διευκολύνει τη σύγκριση των επιδόσεων της νέας γενιάς των cloud data συστημάτων». Αναπτύχθηκε συγκεκριμένα για εργασίες επεξεργασίας συναλλαγών που διέφεραν από αυτές που χρησιμοποιούσαν τα benchmarks που είχαν σχεδιαστεί για παραδοσιακά συστήματα διαχείρισης βάσεων δεδομένων [58].Παρέχει workloads που χρησιμοποιούνται σε μελέτες σύγκρισης για τις NoSQL databases. Έχει διαθέσιμους clients σε όλες τις «γνωστές» NoSQL dbs, όπως Cassandra, DynamoDB, HBase, HyperTable, MongoDB, Redi κλπ.

3.2.3. Workload tests

Στην ορολογία του YCSB, ένα workload ορίζεται ως ένα καθορισμένο σύνολο λειτουργιών που εκτελείται πάνω στη βάση δεδομένων. Τα workloads αποθηκεύονται ως flat αρχεία, και εκτελούνται από συγκεκριμένες κλάσεις που κάνουν extend στην abstract Workload κλάση [59]. Τα workloads δίνουν τη δυνατότητα στον χρήστη να καθορίζει τα ποσοστά Reads vs Writes vs Updates vs Deletes, ώστε να μπορούν να προσαρμοστούν στην εκάστοτε ανάγκη για σύγκριση. Τα workload που χρησιμοποιήθηκαν για να ελεγχθούν και να συγκριθούν οι επιδόσεις των βάσεων παρουσιάζονται στη συνέχεια :

Workload A: Update heavy workload: 50/50% read/update. Ένα παράδειγμα εφαρμογής είναι ένα session store που καταγράφει πρόσφατες ενέργειες [60].

Workload B: Read mostly workload: 95/5% read/update. Παράδειγμα εφαρμογής: photo tagging – η πρόσθεση μιας ετικέτας είναι ένα update, αλλά οι περισότερες ενέργειες εμπίπτουν στο διάβασμα των ετικετών [60].

Workload C: Read only: 100% reads. Παράδειγμα εφαρμογής: user profile cache, όπου τα προφίλ φτιάχνονται αλλού (πχ. Hadoop) [60].

Workload D: Read latest workload: More traffic on recent inserts (95% reads, 5% insert). Σε αυτό το workload προστίθενται νέες εγγραφές και οι πιο πρόσφατα εισαχθέντες είναι οι πιο

δημοφιλείς. Παράδειγμα εφαρμογής: user status updates – οι χρήστες θέλουν να διαβάζουν τις πιο πρόσφατες [60].

Workload E: Short ranges: Short range based queries 95-5% scan-insert. Σε αυτό το workload γίνονται queries πάνω σε short ranges των records. Παράδειγμα εφαρμογής: threaded conversations, όπου κάθε scan είναι για τα posts σε ένα συγκεκριμένο thread [60].

Workload F: Read-modify-write: Read, modify and update existing records 50-50% read/read-modify-write ratio. Σε αυτό το workload, ο χρήστης διαβάζει ένα record, το τροποποιεί και γράφει πίσω τις αλλαγές. Παράδειγμα εφαρμογής: user database [60].

Για το Setup της Cassandra, αρχικά δημιουργήθηκε ένα keyspace με το όνομα “userspace”, και ένα column family με το όνομα “data”.

Για τα τεστ χρησιμοποιήθηκαν οι εξής παράμετροι για τα δεδομένα που εισήχθησαν:

- Recordcount = 10000000 (τα αρχεία περιέχουν ένα πεδίο των 100 bytes - 151 bytes total)
- Operationcount = 100000 (η εκτέλεση των workloads έγινε χρησιμοποιώντας 100000 operations, το οποίο σημαίνει ότι 100000 requests γίνονται στην βάση για κάθε test)
- Request distribution = zipfian

Χρησιμοποιήθηκε επίσης η παράμετρος -target n: αριθμός n λειτουργιών ανά δευτερόλεπτο (ops/sec). Με την παράμετρο αυτή ουσιαστικά ορίζουμε το target throughput. Χωρίς το target θα κανει όσα operations χωράνε σε ένα δευτερόλεπτο, δηλαδή το μέγιστο, ενώ με συγκεκριμένο στόχο στο throughput μπορούμε να ελέγξουμε τι καθυστέρηση έχει κάθε workload για συγκεκριμένο target throughput (latency αυτού του throughput). Στην συγκεκριμένη σειρά πειραμάτων τα target throughput που χρησιμοποιήθηκαν είναι 1000, 5000, 10000, 50000 και 100000.

Για να γίνει η σύγκριση και να είναι το μέγεθος την βάσης συνεπές, ακολουθήθηκε η εξής σειρά (σύμφωνα με την πρόταση του YCSB documentation [60]):

1. Αρχικά φορτώθηκε το σετ δεδομένων του workload A (workloads/workloada) χρησιμοποιώντας το “-load” switch στον client.

2. Εκτελέστηκε το τεστ workload A (χρησιμοποιώντας το workloads/workloada και την παράμετρο “-target”) για μια ποικιλία από throughputs.

3. Εκτελέστηκε το τεστ workload B (χρησιμοποιώντας το workloads/workloadb και την παράμετρο “-target”) για μια ποικιλία από throughputs.

4. Εκτελέστηκε το τεστ workload C (χρησιμοποιώντας το workloads/workloadc και την παράμετρο “-target”) για μια ποικιλία από throughputs.

5. Εκτελέστηκε το τεστ workload F (χρησιμοποιώντας το workloads/workloadf και την παράμετρο “-target”) για μια ποικιλία από throughputs.

6. Εκτελέστηκε το τεστ workload D (χρησιμοποιώντας το workloads/workloadd και την παράμετρο “-target”) για μια ποικιλία από throughputs. Το αρχείο αυτό εισάγει δεδομένα, μεγαλώνοντας το μέγεθος της βάσης.

7. Σβήνουμε τα δεδομένα της βάσης

8. Φορτώθηκε το σετ δεδομένων του workload E (workloads/workloade) χρησιμοποιώντας το “-load” switch στον client.

9. Εκτελέστηκε το τεστ workload E (χρησιμοποιώντας το workloads/workloade και την παράμετρο “-target”) για μια ποικιλία από throughputs.

Το αποτέλεσμα του κάθε τεστ είναι η μέση, η ελάχιστη, η μέγιστη, η 95% και η 99% latency για κάθε είδος λειτουργίας (read, update κλπ), ένας μετρητής των return codes για κάθε λειτουργία και ένα ιστόγραμμα από τα latencies της κάθε λειτουργίας. Το ιστόγραμμα δείχνει τον αριθμό των calls που επέστρεψαν μέσα σε έναν προκαθορισμένο χρονικό διάστημα (milliseconds). Για παράδειγμα:

0 45553 → 45553 calls returned in 0 ms

1 2344 → 2344 calls returned in 1ms

2 399 → 399 calls returned in 2 ms

Ο σύνηθης τρόπος για την αναπαράσταση των αποτελεσμάτων είναι τα διαγράμματα Throughput vs Latency για ένα συγκεκριμένο Workload. Όπου Throughput χρησιμοποιούμε τα προκαθορισμένα target throughputs και όπου Latency τα 95 ή 99% latency ή το average latency .

Workload A

Πίνακας 4: Αποτελέσματα από το Loading των δεδομένων του workload A για την Cassandra και την Mongo

	Cassandra	Mongo
overall		
RunTime(ms)	3265163	2409978
Throughput(ops/sec)	3062.63	4149.42
insert		
Operations	10000000	10000000
AverageLatency(us)	321.59	238.43
MinLatency(us)	101	19
MaxLatency(us)	992989	38631959
95thPercentileLatency(ms)	0	0
99thPercentileLatency(ms)	0	0

Για το φόρτωμα τον δεδομένων του Workload A, με τα οποία έγιναν και τα περισσότερα τεστς, η Cassandra έκανε συνολικό χρόνο 54,5 περίπου λεπτά με απόδοση 3062 oparetions/sec. Για τον ίδιο φόρτο, η Mongo παρουσίασε σχετικά καλύτερα αποτελέσματα με συνολικό χρόνο 40 λεπτά, εκτελώντας 4149 oparetions/sec.

Cassandra → Run workload A

Πίνακας 5: Αποτελέσματα από την Cassandra για το Workload A

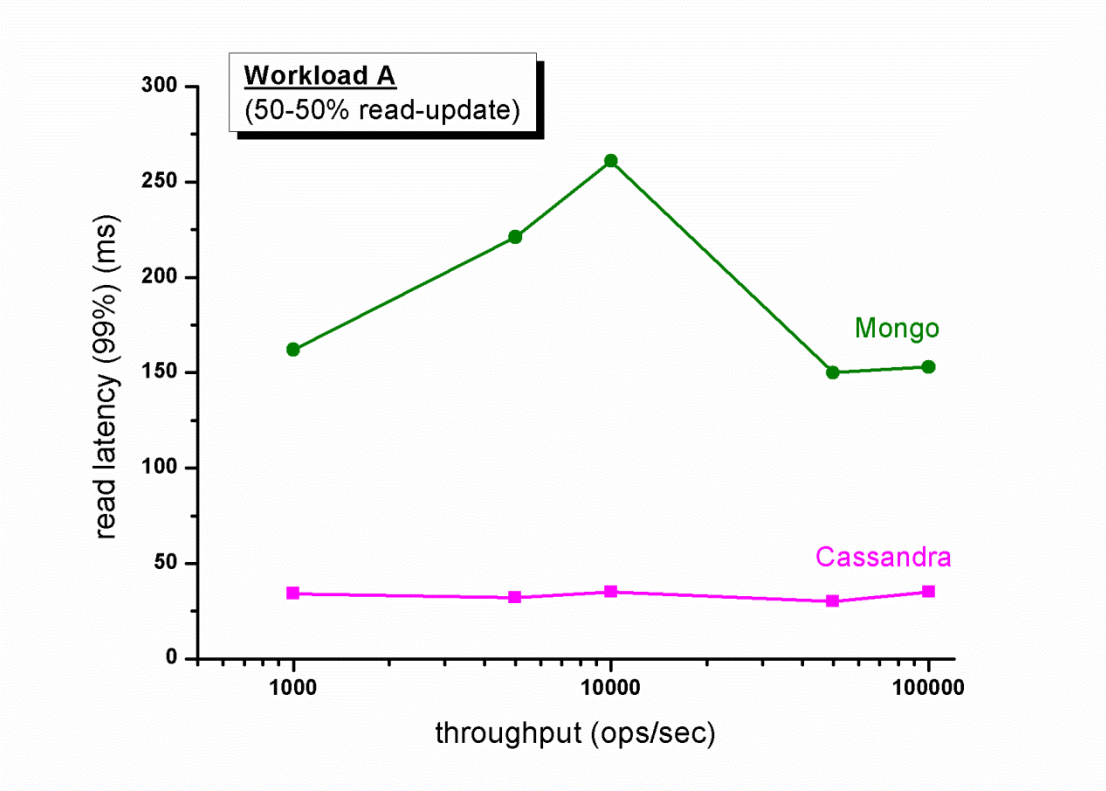
	Target=1000	Target=5000	Target=10000	Target=50000	Target=100000
overall					
RunTime(ms)	368117	358691	372003	339343	386023
Throughput(ops/sec)	271.65	278.79	268.82	294.69	259.05
Read					
Operations	50141	50145	49893	49925	49707
AverageLatency(us)	6972.31	6786.04	7083.36	6425.37	7389.90
MinLatency(us)	126	215	146	225	198
MaxLatency(us)	617185	1037303	1060382	575887	485254
95thPercentileLatency (ms)	20	19	20	19	21
99thPercentileLatency (ms)	34	32	35	30	35
Update					
Operations	49859	49855	50107	50075	50293
AverageLatency(us)	330.34	327.87	330.37	329.51	331.33
MinLatency(us)	76	77	83	82	82
MaxLatency(us)	38532	51497	55624	51899	35462
95thPercentileLatency (ms)	0	0	0	0	0
99thPercentileLatency (ms)	0	0	0	0	0

Mongo → Run workload A

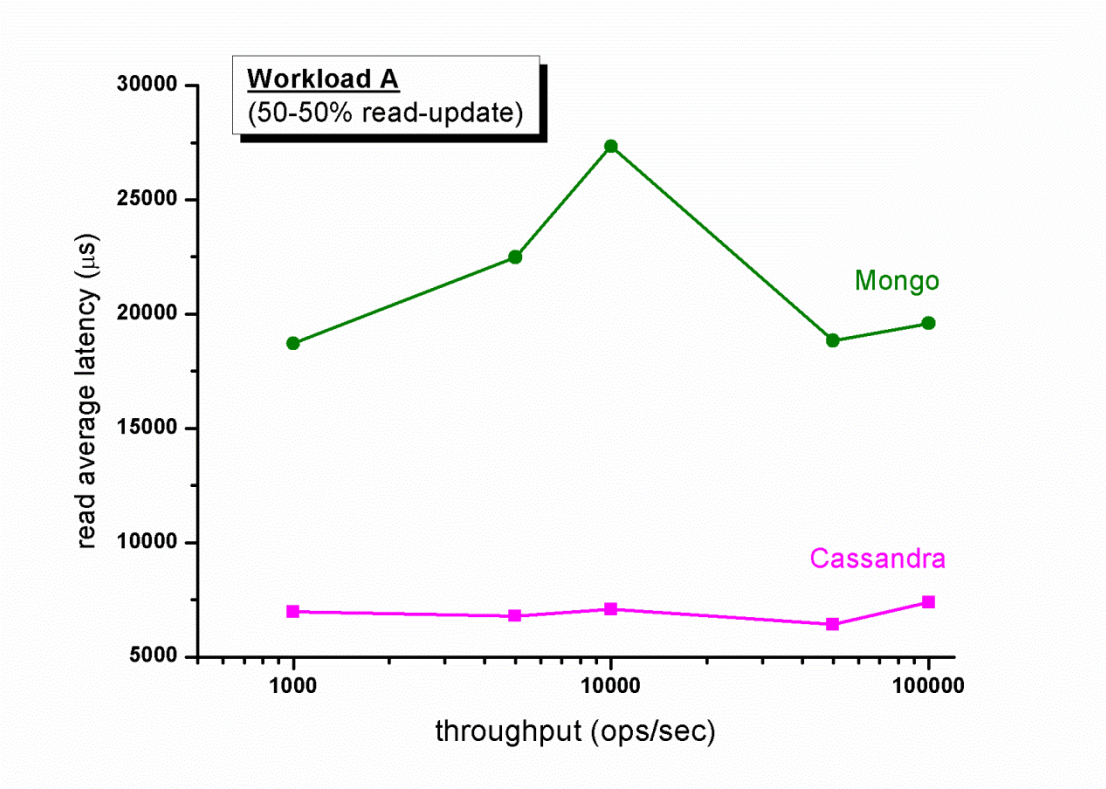
Πίνακας 6: Αποτελέσματα από την Mongo για το Workload A

	Target=1000	**Target=5000**	**Target=10000**	**Target=50000**	**Target=100000**
overall					
RunTime(ms)	947449	1133692	1373278	956083	985036
Throughput(ops/sec)	105.55	88.20	72.82	104.59	101.52
Read					
Operations	50133	49984	49883	50251	49792
AverageLatency(us)	18710.10	22477.02	27333.59	18836.12	19588.80
MinLatency(us)	69	86	82	78	82
MaxLatency(us)	5774830	6225265	10073952	4435793	5201214
95thPercentileLatency (ms)	49	62	88	51	55
99thPercentileLatency (ms)	162	221	261	150	153
Update					
Operations	49867	50016	50117	49749	50208
AverageLatency(us)	148.76	149.05	149.78	150.41	148.49
MinLatency(us)	16	16	16	16	16
MaxLatency(us)	96299	92513	112272	98212	20713
95thPercentileLatency (ms)	0	0	0	0	0
99thPercentileLatency (ms)	1	1	1	1	1

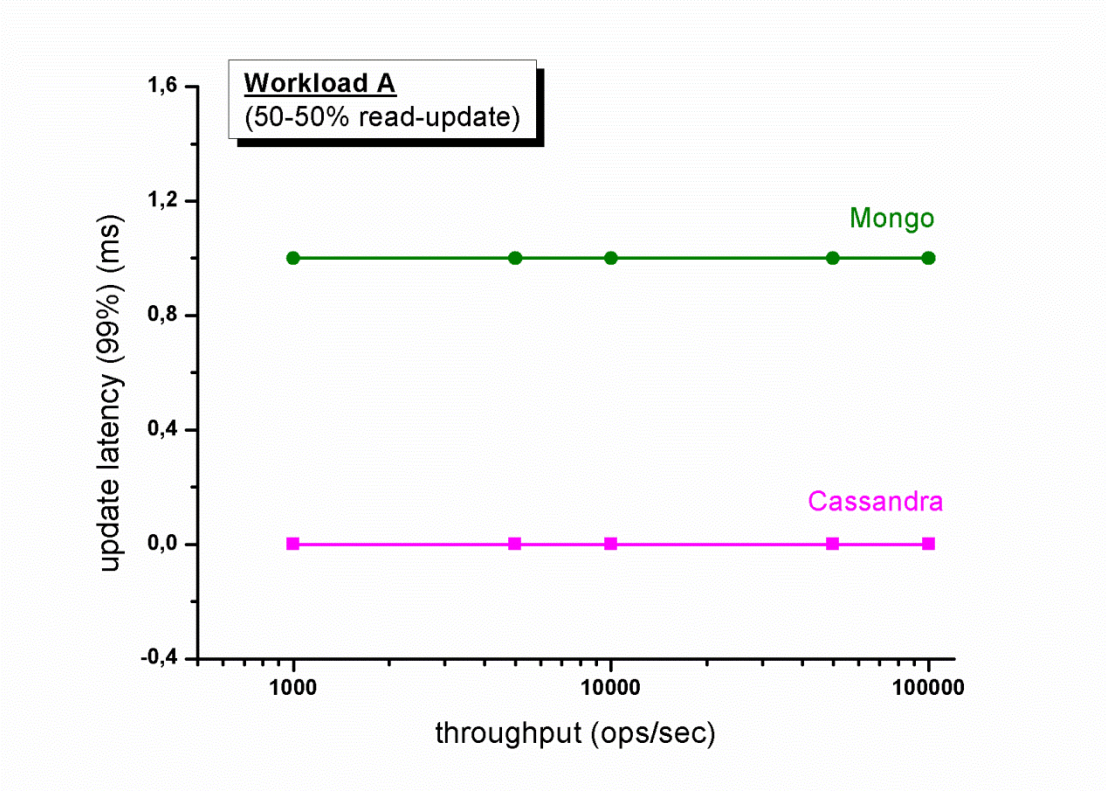
Στο σύνολο του τεστ Workload A η Cassandra παρουσιάζει αρκετά καλύτερα αποτελέσματα. Έχει συνολικό μέσο χρόνο εκτέλεσης τα 6 min, και μέσο συνολικό throughput = 275 ops/sec. Η Mongo αντίστοιχα έχει τριπλάσιο χρόνο εκτέλεσης τα 18 min, και περίπου 3 φορές μικρότερο μέσο συνολικό throughput = 94.4 ops/sec. Όπως φαίνεται και από τις εικόνες 17, 18, 19, 20 που ακολουθούν, η Cassandra έχει μια τάξη μεγέθους χαμηλότερο latency (average και 99th percentile) σε σχέση με τη Mongo, και παρουσιάζει επίσης πιο σταθερή συμπεριφορά ανάλογα με το μέγιστο αριθμό requests που της επιβάλλουμε να κάνει (παράμετρος target → ops/sec).



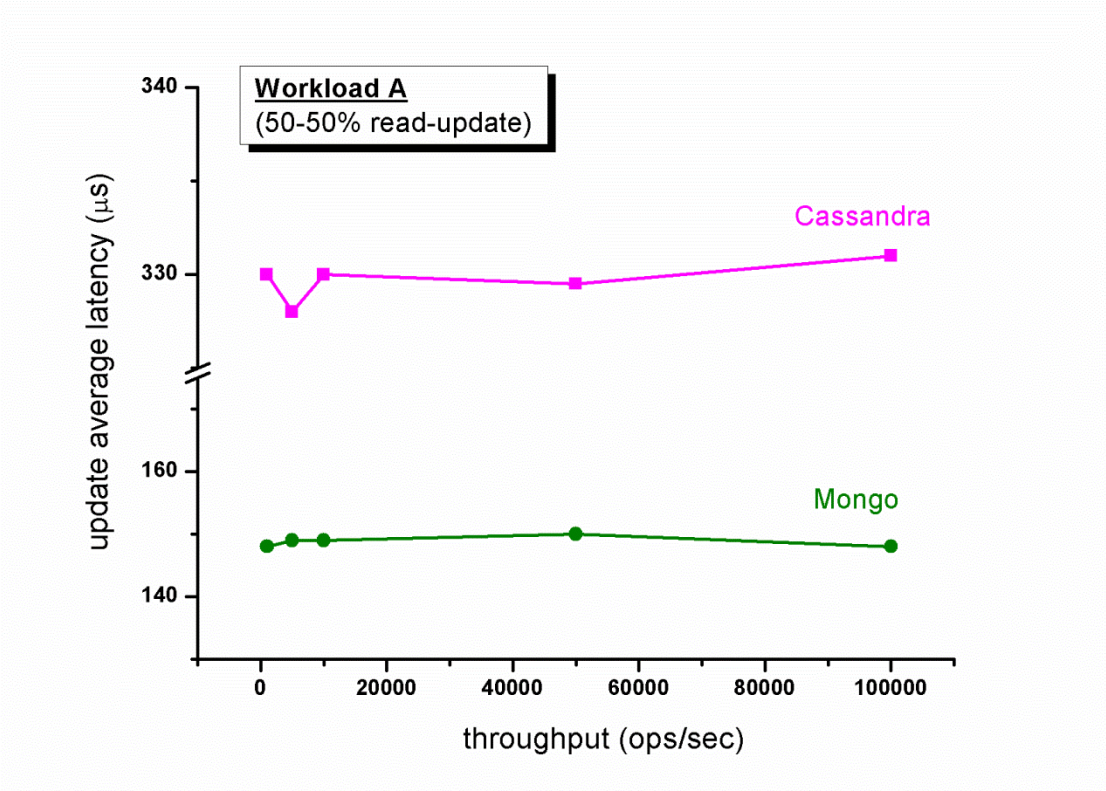
Εικόνα 17: 99th percentile read latency vs throughput για το Workload A



Εικόνα 18: Average read latency vs throughput για το Workload A



Εικόνα 19: 99th percentile update latency vs throughput για το Workload A



Εικόνα 20: Average update latency vs throughput για το Workload A

Workload B

Cassandra → Run workload B

Πίνακας 7: Αποτελέσματα από την Cassandra για το Workload B

	Target=1000	Target=5000	Target=10000	Target=50000	Target=100000
overall					
RunTime(ms)	610117	629438	627055	604884	695714
Throughput(ops/sec)	163.90	158.87	159.47	165.32	143.73
Read					
Operations	95011	95044	95022	95010	95008
AverageLatency(us)	6366.46	6567.94	6543.21	6311.62	6646.95
MinLatency(us)	203	217	131	222	180
MaxLatency(us)	1140029	1285562	662330	1054154	378605
95thPercentileLatency (ms)	18	19	19	18	19
99thPercentileLatency (ms)	31	32	32	30	29
Update					
Operations	4989	4956	4978	4990	4992
AverageLatency(us)	644.47	650.38	656.18	642.66	678.04
MinLatency(us)	137	144	131	202	318
MaxLatency(us)	48656	31754	70413	23950	85854
95thPercentileLatency (ms)	0	0	0	0	1
99thPercentileLatency (ms)	7	7	8	8	7

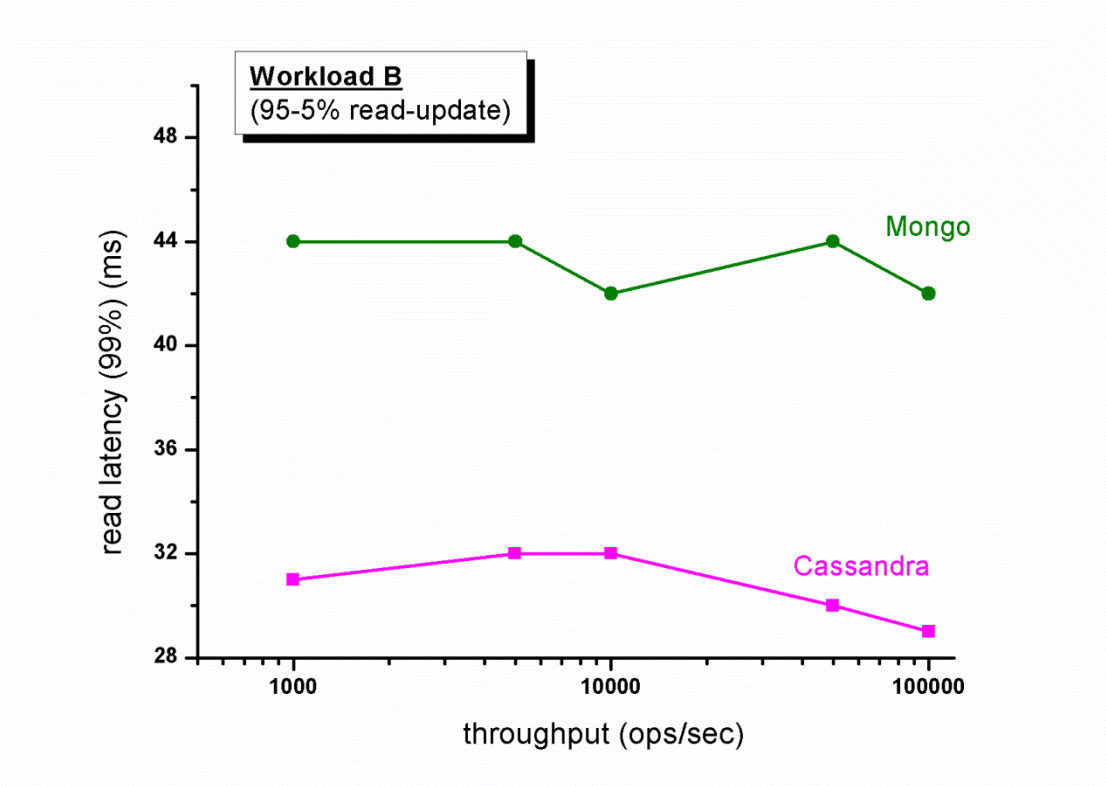
Mongo → Run workload B

Πίνακας 8: Αποτελέσματα από την Mongo για το Workload B

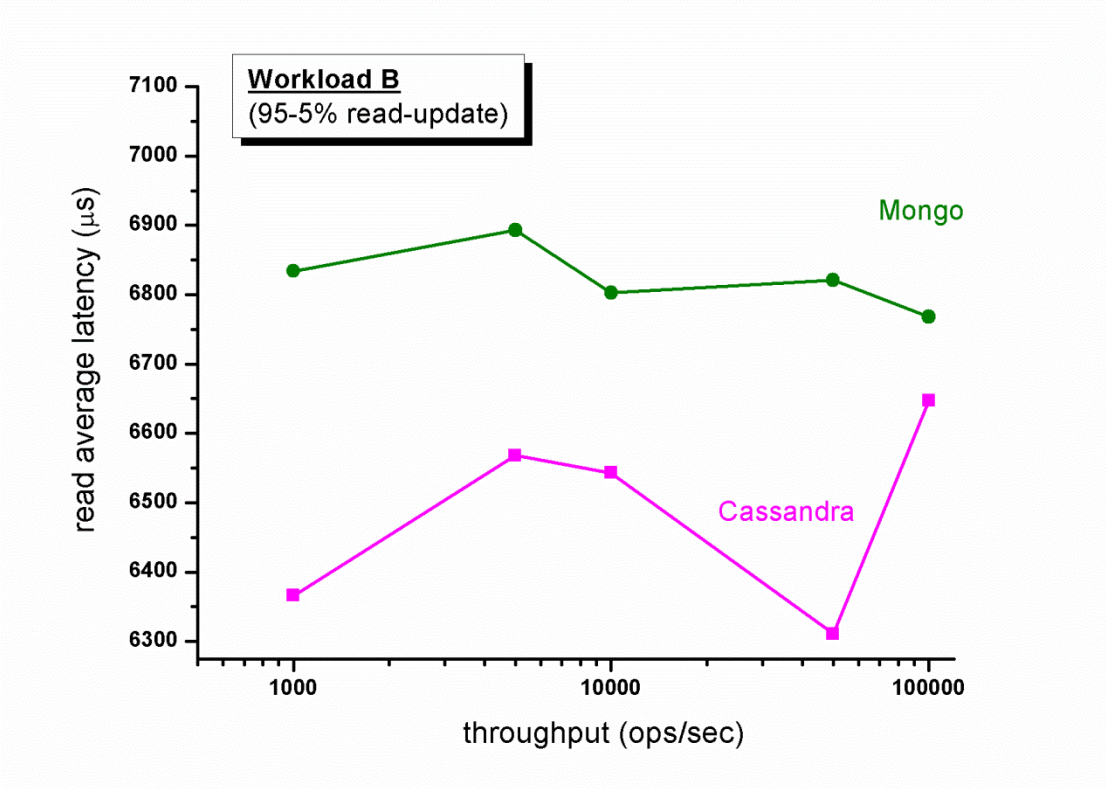
	Target=1000	Target=5000	Target=10000	Target=50000	Target=100000
overall					
RunTime(ms)	653592	658347	651089	651928	646618
Throughput(ops/sec)	153.00	151.89	153.58	153.39	154.65
Read					
Operations	95054	94926	95122	94993	94952
AverageLatency(us)	6834.77	6893.22	6803.26	6821.93	6768.68
MinLatency(us)	66	71	80	69	66
MaxLatency(us)	1783377	1140970	958836	1086229	715785
95thPercentileLatency (ms)	23	23	23	23	23
99thPercentileLatency	44	44	42	44	42

(ms)					
Update					
Operations	4946	5074	4878	5007	5048
AverageLatency(us)	370.42	371.30	385.23	369.92	369.70
MinLatency(us)	84	81	93	88	88
MaxLatency(us)	27149	20312	20223	25953	20676
95thPercentileLatency (ms)	0	0	0	0	0
99thPercentileLatency (ms)	1	1	1	1	1

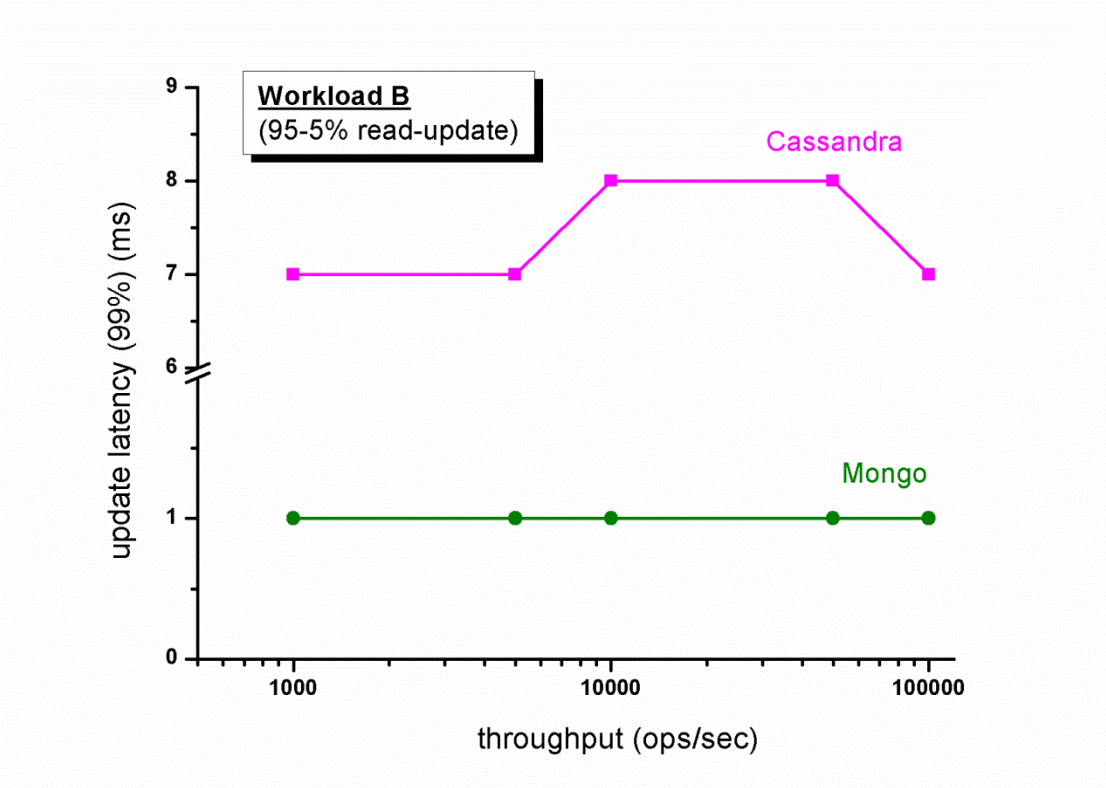
Στο workload B, το οποίο είναι κατά 95% read και κατά 5% write, η Cassandra παρουσίασε αρκετά χαμηλότερο read latency σε σχέση με την Mongo, ενώ στο update latency είχε διπλάσια νούμερα. Σχετικά με τον συνολικό χρόνο του τεστ, το runtime της ήταν 10.5 min με 158.2 operations/sec, ενώ αντίστοιχα για την Mongo μετρήθηκε μέσος χρόνος εκτέλεσης 11 min και συνολικό throughput 152.3 operations/sec.



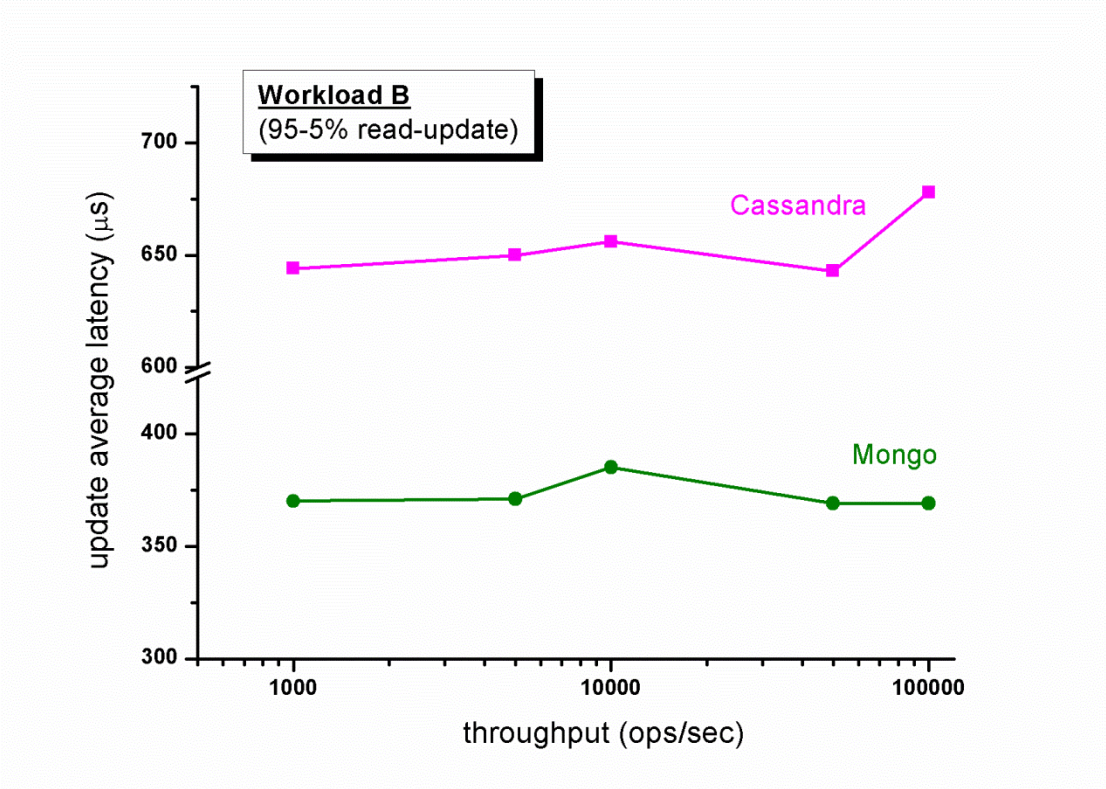
Εικόνα 21: 99th percentile read latency vs throughput για το Workload B



Εικόνα 22: Average read latency vs throughput για το Workload B



Εικόνα 23: 99th percentile update latency vs throughput για το Workload B



Εικόνα 24: Average update latency vs throughput για το Workload B

Workload C

Cassandra → Run workload C

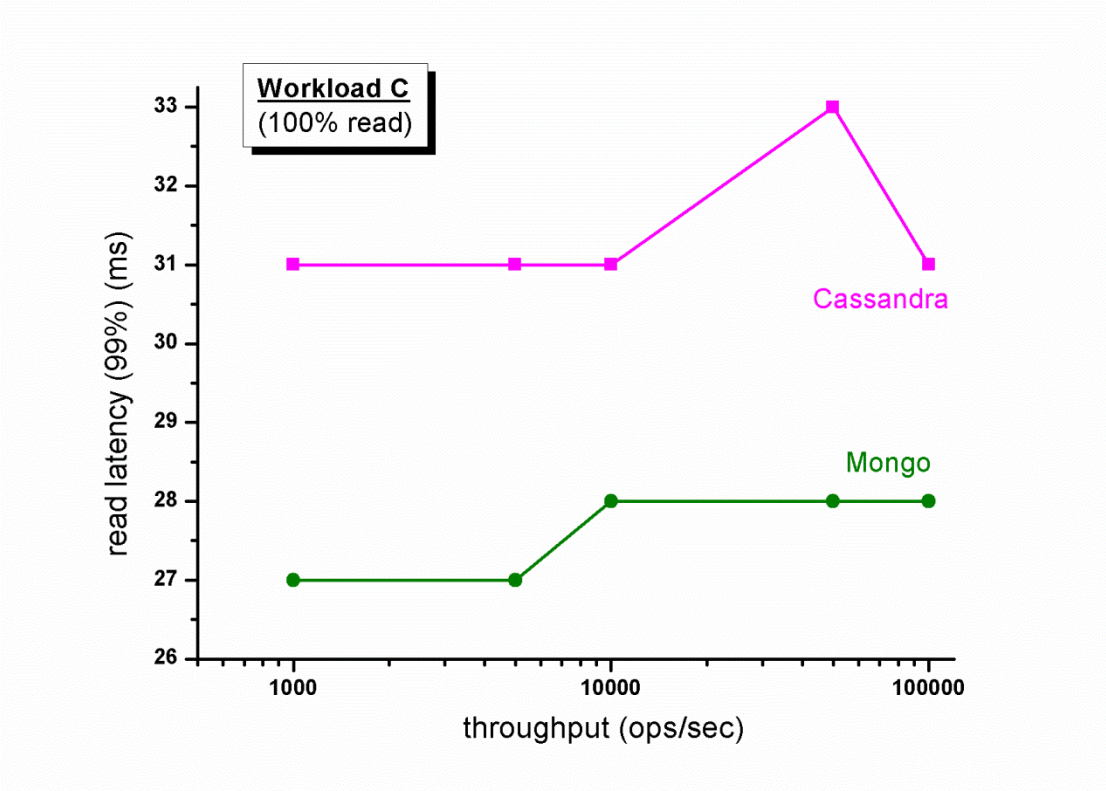
Πίνακας 9: Αποτελέσματα από την Cassandra για το Workload C

	Target=1000	Target=5000	Target=10000	Target=50000	Target=100000
overall					
RunTime(ms)	610117	656920	651889	645754	652897
Throughput(ops/sec)	163.90	152.23	153.40	154.86	153.16
Read					
Operations	100000	100000	100000	100000	100000
AverageLatency(us)	6484.59	6551.84	6501.33	6440.33	6511.58
MinLatency(us)	221	202	199	189	196
MaxLatency(us)	839571	1063276	1409278	733035	1154743
95thPercentileLatency (ms)	19	19	19	19	18
99thPercentileLatency (ms)	31	31	31	33	31

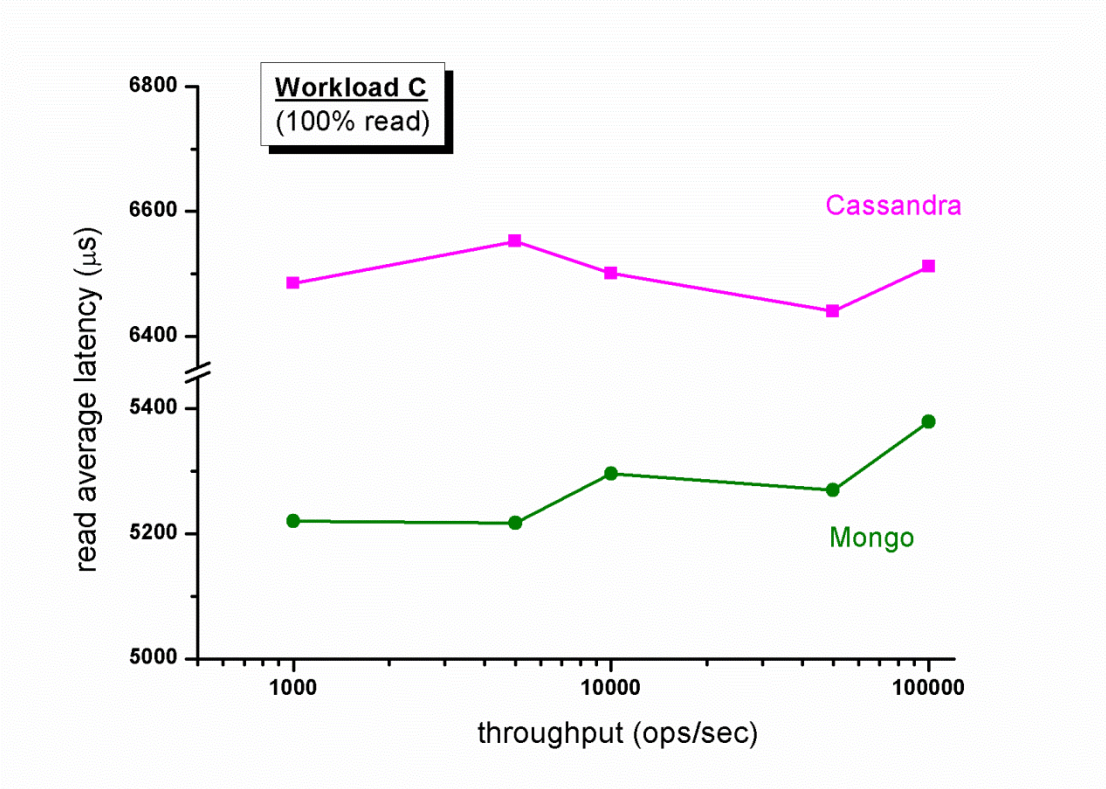
Mongo → Run workload C

Πίνακας 10: Αποτελέσματα από την Mongo για το Workload C

	Target=1000	Target=5000	Target=10000	Target=50000	Target=100000
overall					
RunTime(ms)	524087	523600	531523	529149	539821
Throughput(ops/sec)	190.81	190.98	188.14	188.98	185.25
Read					
Operations	100000	100000	100000	100000	100000
AverageLatency(us)	5220.8	5217.33	5296.87	5270.98	5379.00
MinLatency(us)	67	71	63	64	67
MaxLatency(us)	682737	985964	754889	782450	736372
95thPercentileLatency (ms)	14	14	14	14	14
99thPercentileLatency (ms)	27	27	28	28	28



Εικόνα 25: 99th percentile read latency vs throughput για το Workload C



Εικόνα 26: Average read latency vs throughput για το Workload C

Το workload C αποτελείται από 100% read requests. Η Mongo παρουσιάζει πολύ καλύτερη συμπεριφορά σε σχέση με την Cassandra, όσον αφορά το επιθυμητό low read latency, τον συνολικό χρόνο εκτέλεσης του τεστ (η Cassandra περίπου 11 min, ενώ η Mongo περίπου 9 min) αλλά και το Throughput (155.4 ops/sec για την Cassandra, ενώ 188.8 ops/sec για την Mongo). Αυτό οφείλεται στο ότι η Cassandra σχεδιαστικά είναι βελτιστοποιημένη για εγγραφές (writes) και όχι read, αν και σύμφωνα με τα αποτελέσματα δεν μένει πολύ πίσω από την Mongo.

Workload F

Cassandra → Run workload F

Πίνακας 11: Αποτελέσματα από την Cassandra για το Workload F

	Target=1000	Target=5000	Target=10000	Target=50000	Target=100000
overall					
RunTime(ms)	668506	662743	655627	620257	661600
Throughput(ops/sec)	149.59	150.88	152.53	161.22	151.15
Read					
Operations	100000	100000	100000	100000	100000
AverageLatency(us)	6473.14	6416.09	6343.71	5992.35	6407.27

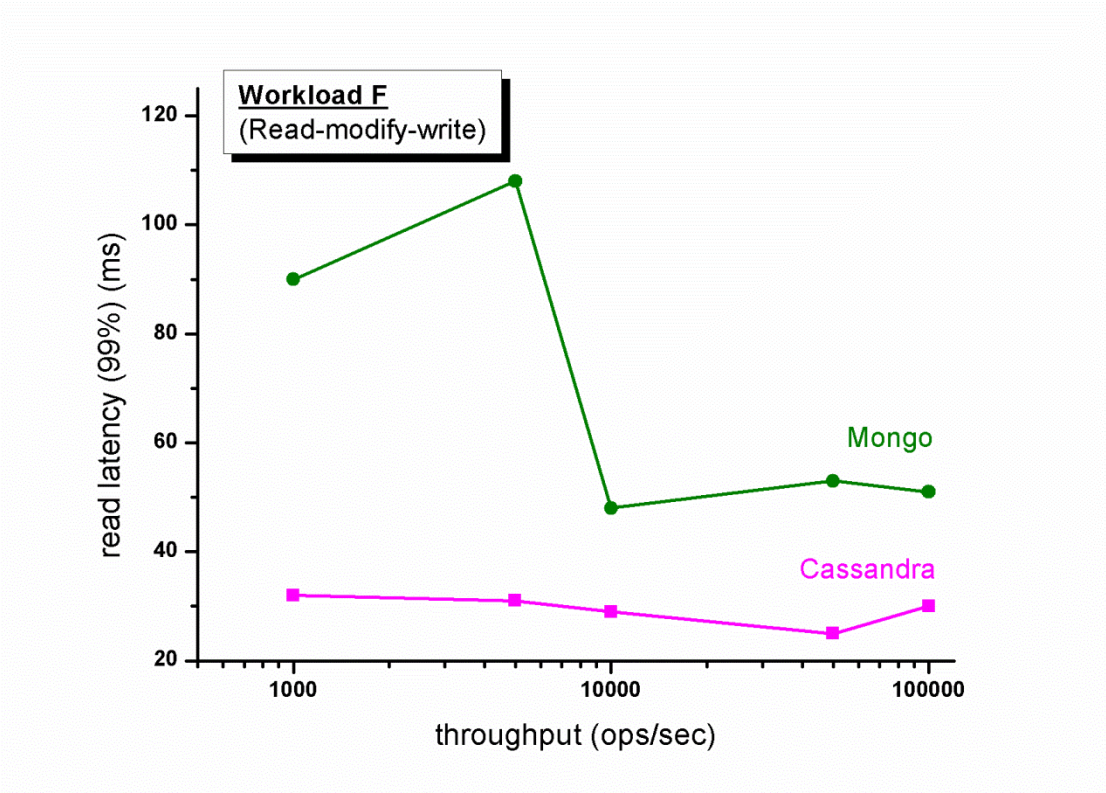
MinLatency(us)	209	139	210	174	146
MaxLatency(us)	1121859	1370867	858543	213964	819324
95thPercentileLatency (ms)	19	19	18	17	19
99thPercentileLatency (ms)	32	31	29	25	30
Read – Modify – Write					
Operations	49959	50049	49852	49935	49733
AverageLatency(us)	6864.93	6825.12	6714.93	6364.72	6794.14
MinLatency(us)	355	361	391	313	296
MaxLatency(us)	1122373	1371338	850984	214312	819925
95thPercentileLatency (ms)	19	19	19	18	19
99thPercentileLatency (ms)	32	32	30	26	31
Update					
Operations	49959	50049	49852	49935	49733
AverageLatency(us)	373.95	372.71	378.78	371.12	370.32
MinLatency(us)	109	107	107	122	106
MaxLatency(us)	52348	113510	338853	50287	42871
95thPercentileLatency (ms)	0	0	0	0	0
99thPercentileLatency (ms)	0	0	0	0	0

Mongo → Run workload F

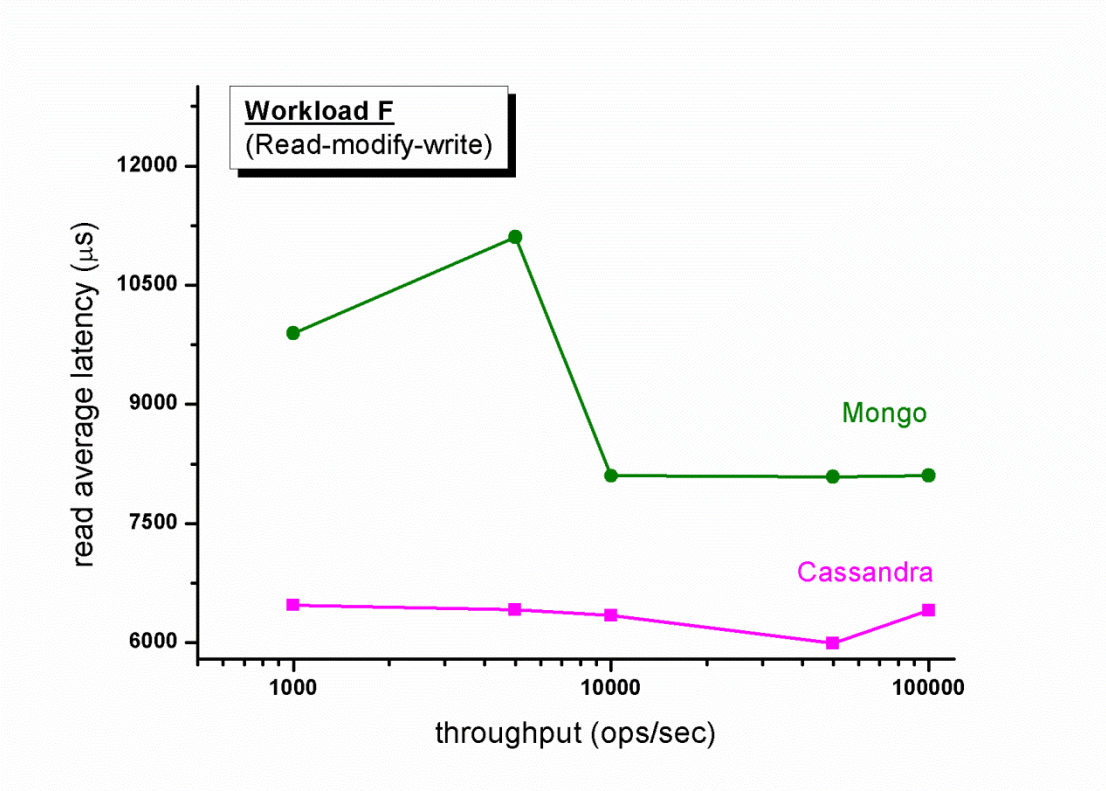
Πίνακας 12: Αποτελέσματα από την Mongo για το Workload F

	Target=1000	**Target=5000**	**Target=10000**	**Target=50000**	**Target=100000**
overall					
RunTime(ms)	999688	1120724	820450	818995	820537
Throughput(ops/sec)	100.03	89.23	121.88	122.10	121.87
Read					
Operations	100000	100000	100000	100000	100000
AverageLatency(us)	9896.33	11105.86	8102.95	8089.48	8104.78
MinLatency(us)	72	67	75	81	73
MaxLatency(us)	4622474	4371509	2250022	2203164	2308069
95thPercentileLatency (ms)	29	30	27	27	27
99thPercentileLatency (ms)	90	108	48	53	51
Read – Modify – Write					
Operations	49898	49917	49838	49788	50046
AverageLatency(us)	10007.29	11145.90	8230.61	8324.82	8293.90
MinLatency(us)	99	97	108	106	97
MaxLatency(us)	4042411	4371639	2250305	2203307	1808583

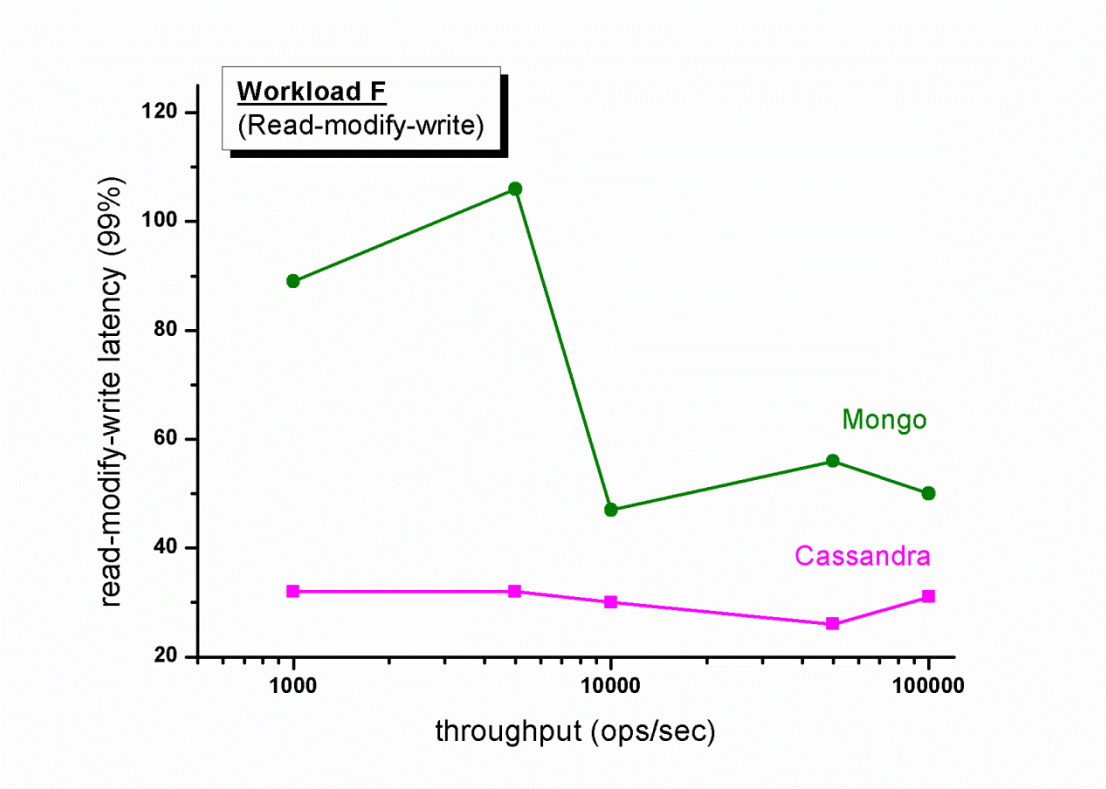
95thPercentileLatency (ms)	29	30	27	27	28
99thPercentileLatency (ms)	89	106	47	56	50
Update					
Operations	49898	49917	49838	49788	50046
AverageLatency(us)	155.12	154.21	153.98	155.33	154.14
MinLatency(us)	20	20	19	20	20
MaxLatency(us)	25534	21528	24275	20241	22178
95thPercentileLatency (ms)	0	0	0	0	0
99thPercentileLatency (ms)	1	1	1	1	1



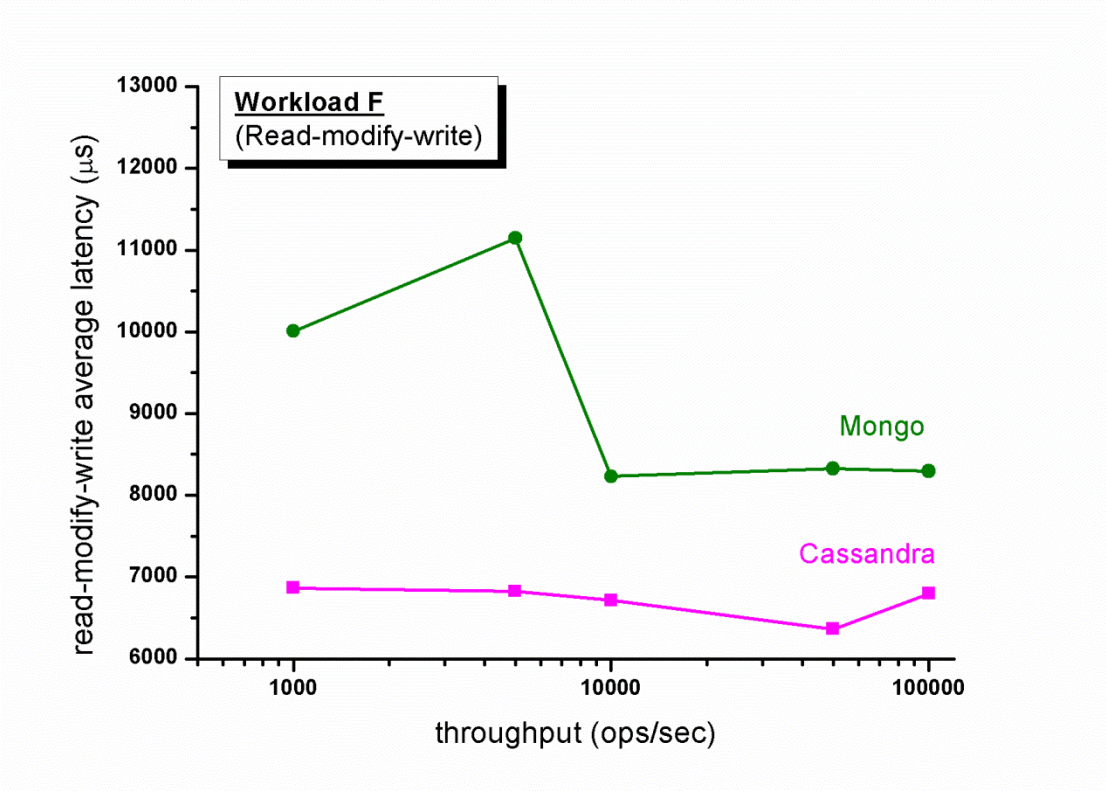
Εικόνα 27: 99th percentile read latency vs throughput για το Workload F



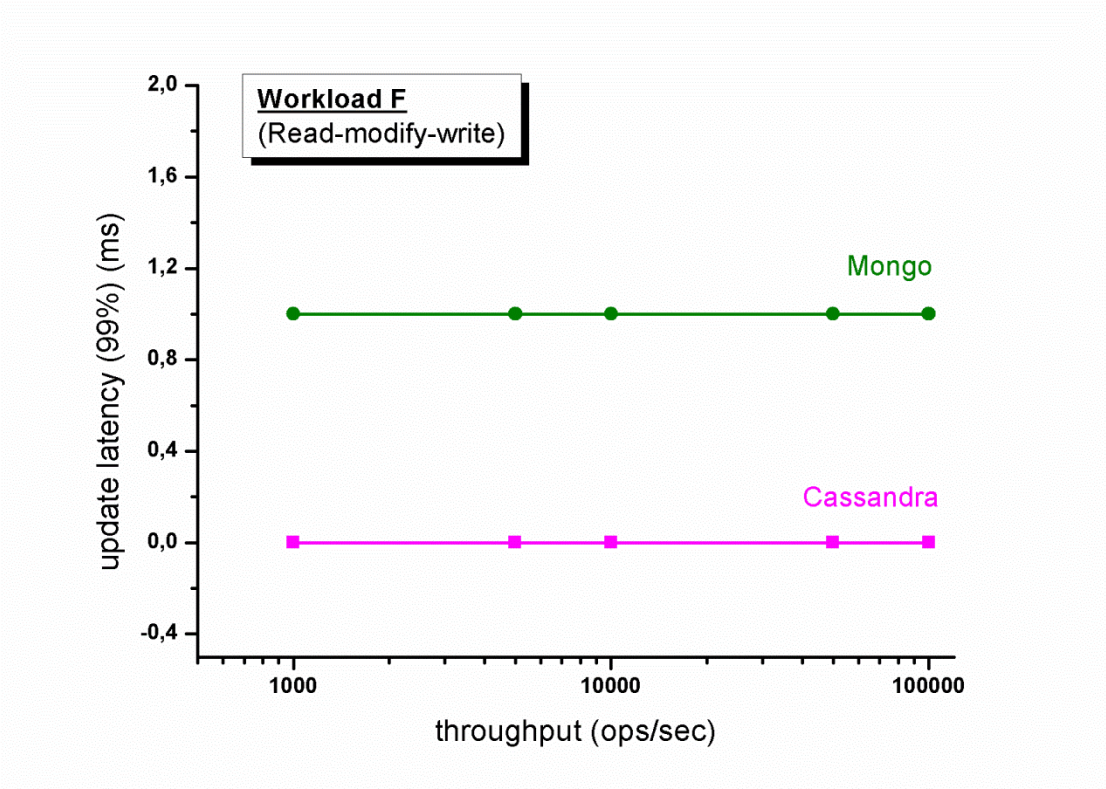
Εικόνα 28: Average read latency vs throughput για το Workload F



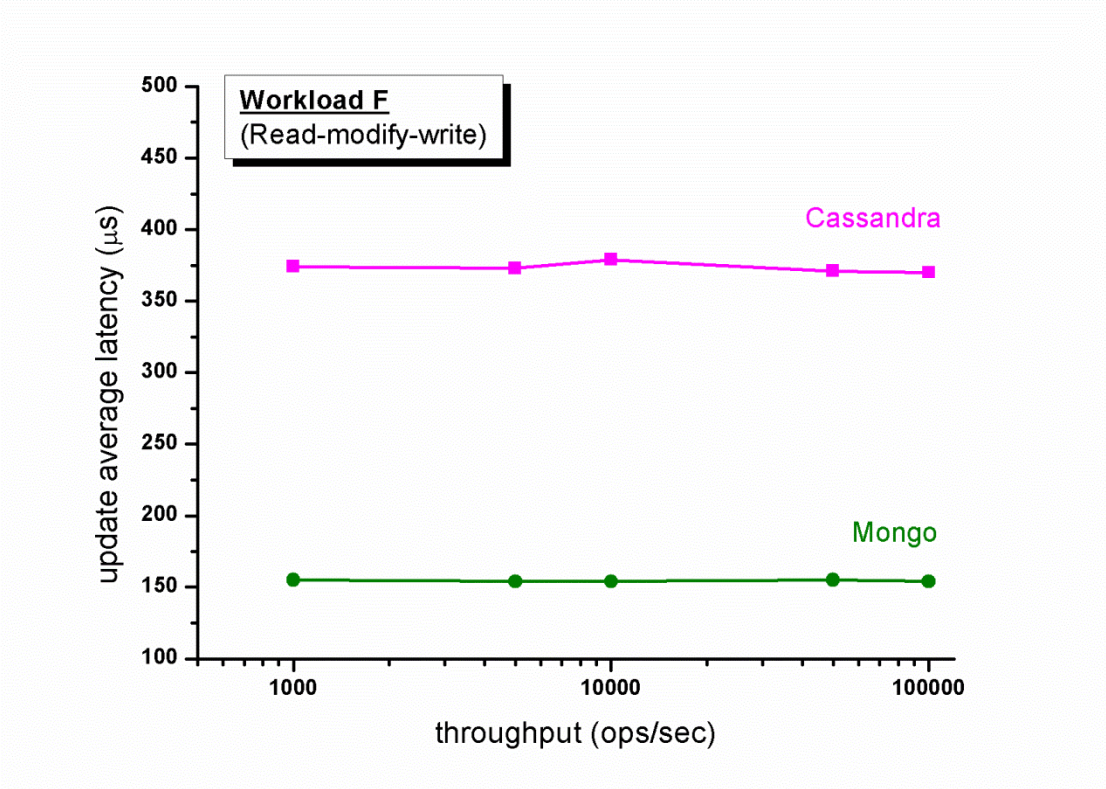
Εικόνα 29: 99th percentile read-modify-write latency vs throughput για το Workload F



Εικόνα 30: Average read-modify-write latency vs throughput για το Workload F



Εικόνα 31: 99th percentile update latency vs throughput για το Workload F



Εικόνα 32: Average update latency vs throughput για το Workload F

Το workload F χρησιμοποιεί 50% read operations και 50% read-modify-write operations. Στατιστικά στο σύνολο του τεστ, η Cassandra έκανε χρόνο εκτέλεσης 11min με μέση απόδοση 153 operations/sec, ενώ αντίστοιχα η Mongo έκανε 15min με μέση απόδοση 111 operations/sec. Τα latencies και για τις 3 φάσεις του τεστ, ήταν πολύ χαμηλότερα για την περίπτωση της Cassandra, σχεδόν με μισές τιμές.

Workload D

Cassandra → Run workload D

Πίνακας 13: Αποτελέσματα από την Cassandra για το Workload D

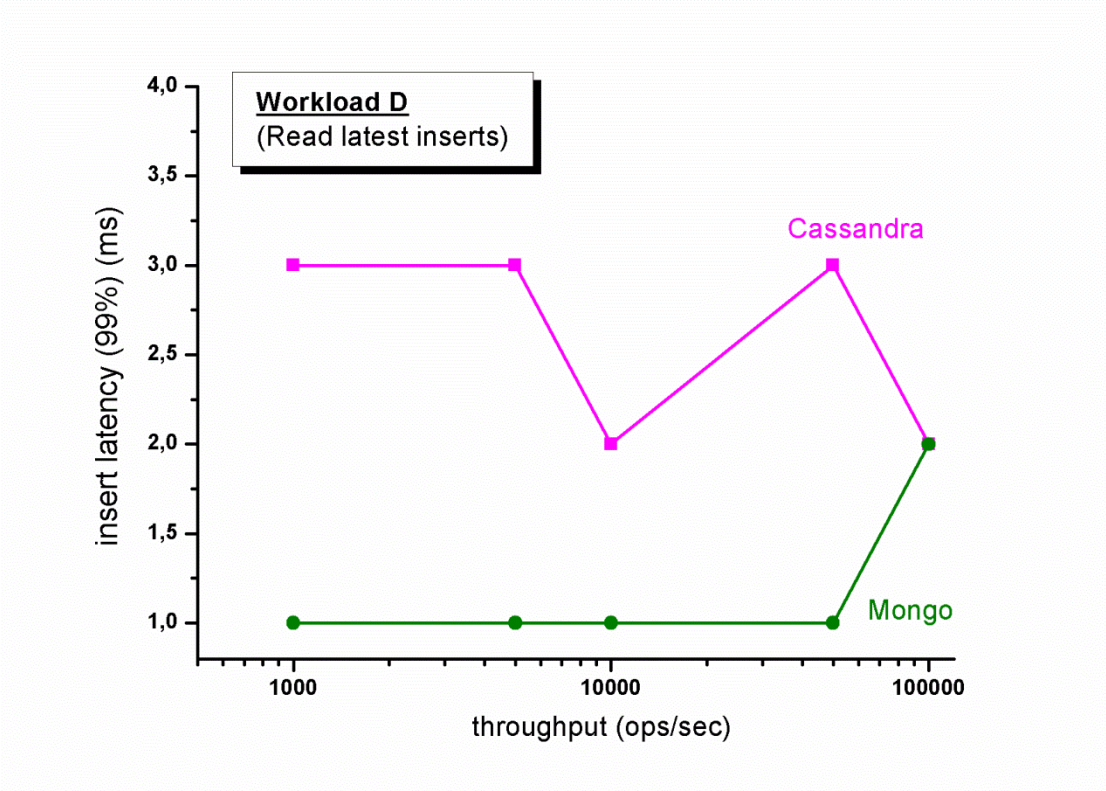
	Target=1000	Target=5000	Target=10000	Target=50000	Target=100000
overall					
RunTime(ms)	256522	226733	214443	204302	202745
Throughput(ops/sec)	389.83	441.05	466.32	489.47	493.23
Insert					
Operations	5022	4933	5009	4990	5143
AverageLatency(us)	633.14	651.30	600.37	640.26	592.10
MinLatency(us)	153	165	166	160	157
MaxLatency(us)	96133	111605	23243	84566	35707

95thPercentileLatency (ms)	1	1	1	1	1
99thPercentileLatency (ms)	3	3	2	3	2
Read					
Operations	94978	95067	94991	95010	94857
AverageLatency(us)	2650.45	2334.22	2208.47	2099.19	2087.79
MinLatency(us)	96	112	104	127	122
MaxLatency(us)	762310	673268	451678	487034	446288
95thPercentileLatency (ms)	13	13	12	12	12
99thPercentileLatency (ms)	23	22	22	21	21

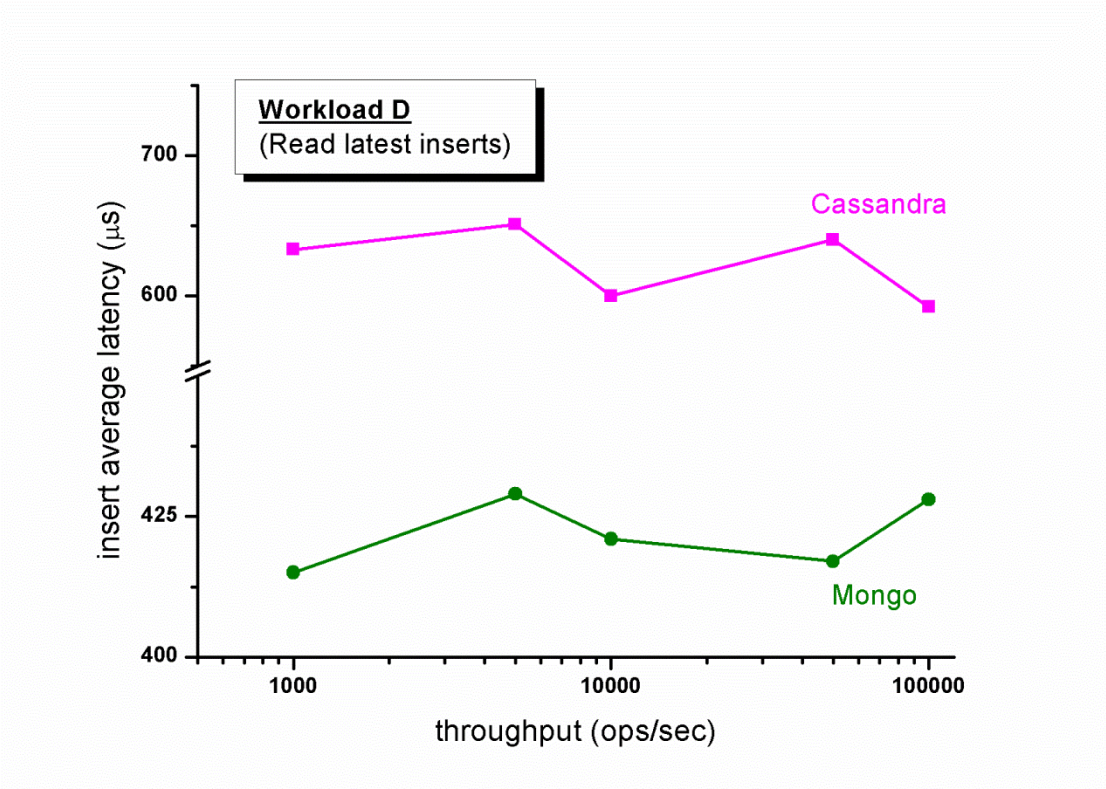
Mongo → Run workload D

Πίνακας 14: Αποτελέσματα από την Mongo για το Workload D

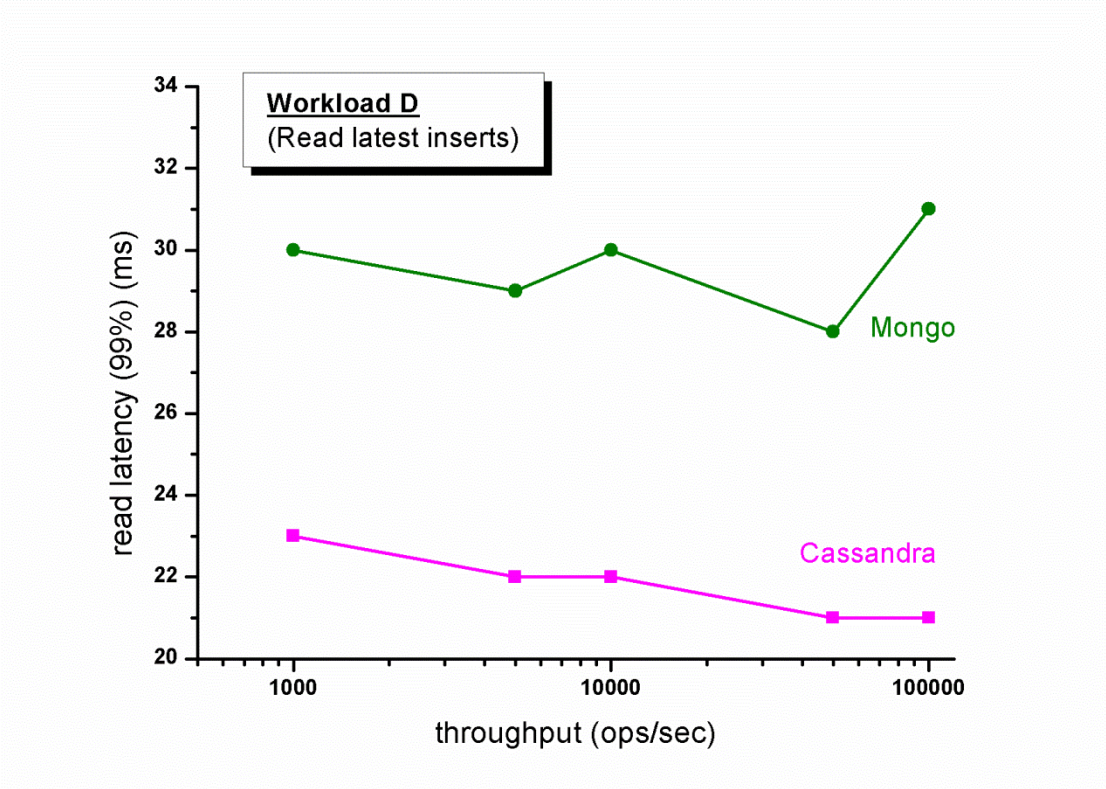
	Target=1000	**Target=5000**	**Target=10000**	**Target=50000**	**Target=100000**
overall					
RunTime(ms)	215853	208900	218740	194390	261222
Throughput(ops/sec)	463.28	478.70	457.16	514.43	382.82
Insert					
Operations	4956	5010	4848	4987	4997
AverageLatency(us)	414.98	429.52	421.28	417.64	427.90
MinLatency(us)	84	86	83	83	91
MaxLatency(us)	25515	22297	29146	82034	22373
95thPercentileLatency (ms)	0	0	0	0	0
99thPercentileLatency (ms)	1	1	1	1	2
Read					
Operations	95044	94990	95152	95013	95003
AverageLatency(us)	2232.93	2160.28	2261.25	2007.75	2707.79
MinLatency(us)	59	59	62	60	65
MaxLatency(us)	293335	407436	284118	398781	1484392
95thPercentileLatency (ms)	12	12	12	12	13
99thPercentileLatency (ms)	30	29	30	28	31



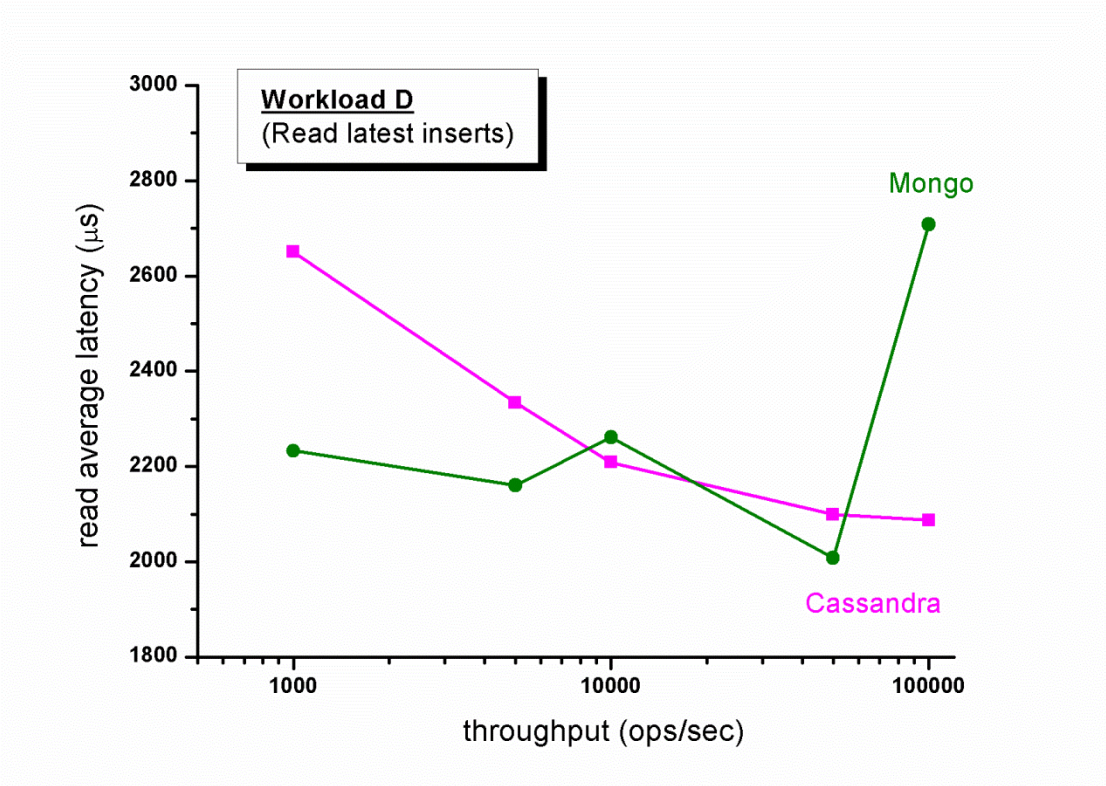
Εικόνα 33: 99th percentile insert latency vs throughput για το Workload D



Εικόνα 34: Average insert latency vs throughput για το Workload D



Εικόνα 35: 99th percentile read latency vs throughput για το Workload D



Εικόνα 36: Average read latency vs throughput για το Workload D

Το workload D έχει βασιστεί σε ένα social network, όπου οι χρήστες στέλνουν διαρκώς status updates με σκοπό να διαβαστούν από άλλλους χρήστες. Το τεστ αυτό αποτελείται από 95% reads και 5% writes, όπως και το workload B, με την διαφορά βέβαια ότι νέα αντικείμενα δημιουργούνται on the fly (Insert vs update) και τα νέα αυτά αντικείμενα είναι αυτά που θα διαβαστούν. Σε αυτό το τεστ, στα συνολικά αποτελέσματα δεν παρατηρούνται ιδιαίτερες διαφοροποιήσεις μεταξύ των δύο βάσεων. Η Cassandra έχει συνολικό χρόνο γύρω στα 3,7min με μέσο throughput = 455.8 operations/sec. Τον ίδιο συνολικό χρόνο έχει και η Mongo με μιά ελάχιστη διαφορά στο throughput που έχει τιμή 459,2 operations/sec. Σχετικά με το latency η Cassandra επιδεικνύει υψηλότερο Insert latency και χαμηλότερο read latency σε σχέση με τη Mongo.

Workload E

Load dataset E

Πίνακας 15: Αποτελέσματα από το Loading των δεδομένων του workload E για την Cassandra και την Mongo

	Cassandra	**Mongo**
overall		
RunTime(ms)	3084524	2435226
Throughput(ops/sec)	3241.99	4106.39
insert		
Operations	10000000	10000000
AverageLatency(us)	303.51	240.61
MinLatency(us)	103	19
MaxLatency(us)	777795	47144502
95thPercentileLatency(ms)	0	0
99thPercentileLatency(ms)	0	0

Για το Workload E δημιουργήθηκε η ανάγκη επαναφόρτωσης των δεδομένων για να υπάρχει consistency στη βάση, λόγω των διαφόρων πράξεων που έγιναν σε αυτή από το τελευτάιο Workload (D). Στο ανέβασμα των δεδομένων η Mongo (~41min) έκανε 10 λεπτά λιγότερο από την Cassandra (51 min), με μέσο throughput = 4106 Operations/sec, συγκριτικά με τα 3242 Operations/sec της Cassandra.

Cassandra → Run workload E

Πίνακας 16: Αποτελέσματα από την Cassandra για το Workload E

	Target=1000	**Target=5000**	**Target=10000**	**Target=50000**	**Target=100000**

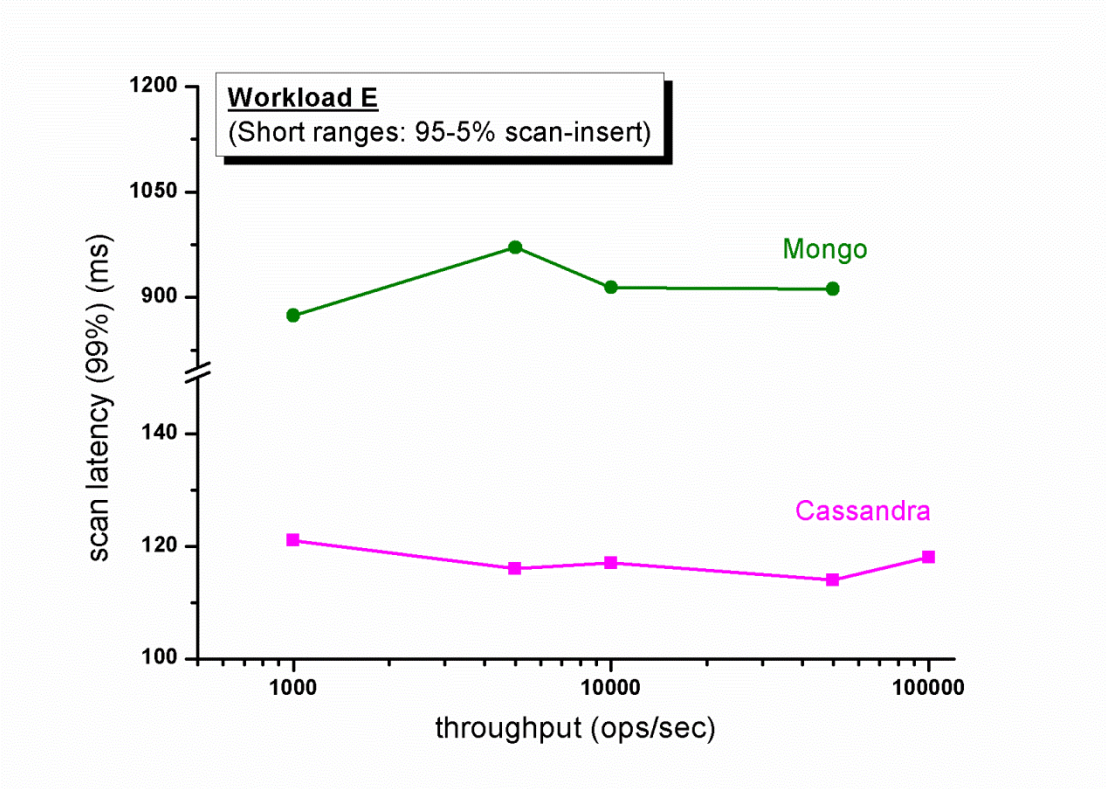
overall					
RunTime(ms)	2721283	2476158	2448943	2435077	2448152
Throughput(ops/sec)	36.75	40.39	40.83	41.07	40.85
Scan					
Operations	94948	95041	95056	94892	94935
AverageLatency(us)	28568.06	25939.70	25657.01	25553.84	25685.81
MinLatency(us)	1708	1831	1765	1627	1724
MaxLatency(us)	1579465	1965106	1982760	1915302	1972666
95thPercentileLatency (ms)	65	60	59	58	58
99thPercentileLatency (ms)	121	116	117	114	118
Insert					
Operations	5052	4959	4944	5108	5065
AverageLatency(us)	1105.68	1546.72	1400.68	1398.62	1305.56
MinLatency(us)	172	228	182	220	234
MaxLatency(us)	174619	189066	231745	429011	33299
95thPercentileLatency (ms)	1	5	5	3	3
99thPercentileLatency (ms)	13	14	14	14	14

Mongo → Run workload E

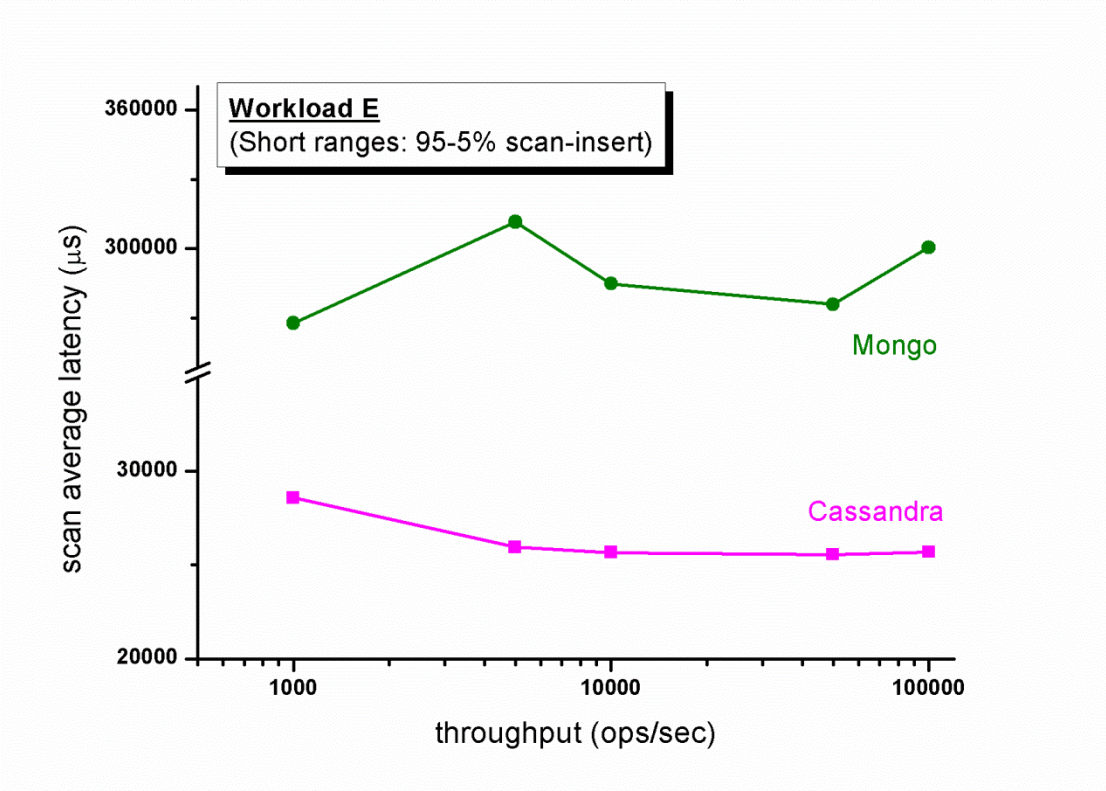
Πίνακας 17: Αποτελέσματα από την Mongo για το Workload E

	Target=1000	**Target=5000**	**Target=10000**	**Target=50000**	**Target=100000**
overall					
RunTime(ms)	2.5463426E7	2.9622419E7	2.7091194E7	2.6196604E7	2.8552202E7
Throughput(ops/sec)	3.93	3.38	3.69	3.82	3.50
Scan					
Operations	95001	95050	95036	94883	94964
AverageLatency(us)	267969.42	311577.06	284995.33	276028.79	300594.36
MinLatency(us)	282	347	240	296	216
MaxLatency(us)	2015099	2592779	1820215	3094346	3592194
95thPercentileLatency (ms)	699	778	714	704	744
99thPercentileLatency (ms)	874	971	914	912	
Insert					
Operations	4999	4950	4964	5117	5036
AverageLatency(us)	625.41	633.06	628.94	623.56	633.10
MinLatency(us)	202	199	216	196	195
MaxLatency(us)	20688	21813	74874	21991	21683
95thPercentileLatency	0	0	0	0	0

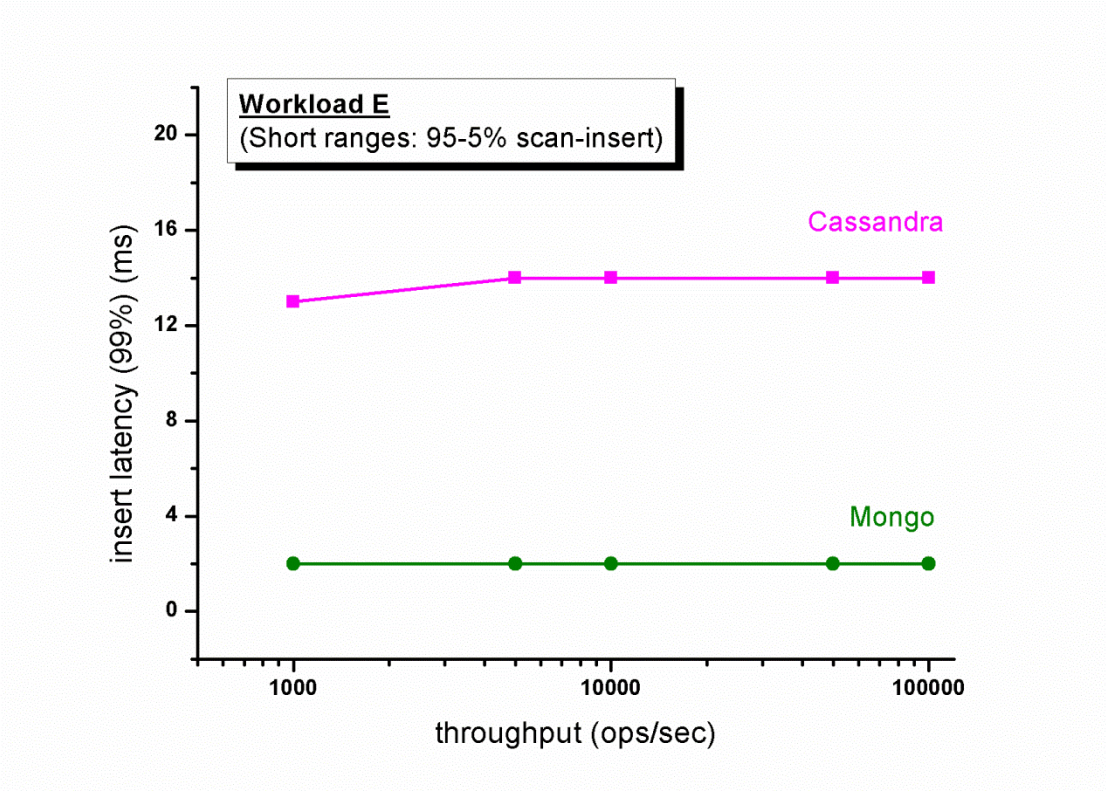
(ms)					
99thPercentileLatency (ms)	2	2	2	2	2



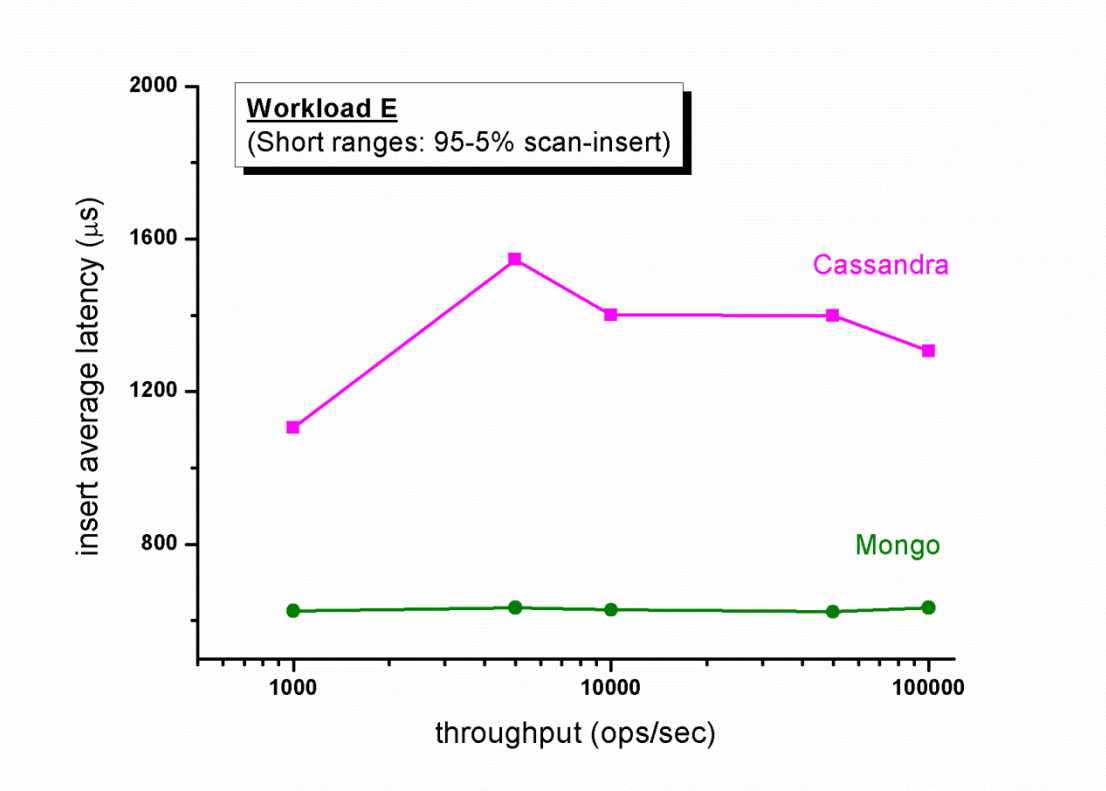
Εικόνα 37: 99th percentile scan latency vs throughput για το Workload E



Εικόνα 38: Average scan latency vs throughput για το Workload E



Εικόνα 39: 99th percentile insert latency vs throughput για το Workload E



Εικόνα 40: Average insert latency vs throughput για το Workload E

Το Workload E βασίζεται σε ένα Online forum με threaded conversations. Αποτελείται κατά 95% από scan operations που εκτελούν range search. Η Cassandra στο τεστ αυτό παρουσιάζει αρκετά καλύτερα συμπεριφορά σε σχέση με τη Mongo, σχεδόν σε όλους τους τομείς. Ο συνολικός της χρόνος για την εκτέλεση του τεστ είναι περίπου 42min με 40 operations/sec, ενώ η Mongo ολοκλήρωσε το τεστ αντίστοιχα σε 7,5 ώρες με 3,7 operations/sec.

4. Επίλογος - Συμπεράσματα

Στην σημερινή εποχή, οι βάσεις δεδομένων θεωρούνται ζωτικής σημασίας μέρος πολλών εταιρειών, οργανισμών και γενικότερα συστημάτων σε όλο τον κόσμο. Μέχρι τώρα, οι σχεσιακές βάσεις δεδομένων ήταν η κατεξοχήν βέλτιστη επιλογή των επιχειρήσεων. Αυτές επιτρέπουν την αποθήκευση, την εξαγωγή και γενικότερα την διαχείρηση των δεδομένων χρησιμοποιώντας μια πρότυπη γλώσσα επερωτήσεων, την SQL. Ωστόσο με τη συνεχή αύξηση του όγκου των δεδομένων προς αποθήκευση και επεξεργασία, οι σχεσιακές βάσεις παρουσιάζουν μια ποικιλία περιορισμών. Οι περιορισμοί αυτοί περιλαμβάνουν την δυσκολία κλιμάκωσης και αποθήκευσης λόγω του μεγάλου όγκου των δεδομένων (μπορούν να φτάσουν μέχρι και τάξη peta-bytes). Για να ξεπεραστούν οι περιορισμοί αυτοί, αναπτύχθηκε ένα νέο μοντέλο βάσεων δεδομένων, με μια σειρά από νέο χαρακτηριστικά, το οποίο ονομάζεται NoSQL. Τα συστήματα αυτά είναι οριζόντια επεκτάσιμα, κατανεμημένα και μη σχεσιακά. Μπορούν να διαχειρίζονται μεγάλους όγκους αδόμητων ή χαλαρά δομημένων δεδομένων. Οι μη σχεσιακές βάσεις δεδομένων αναδείχθηκαν ως μια πρωτοποριακή τεχνολογία και μπορούν να χρησιμοποιηθούν μόνες ή σαν συμπλήρωμα στις σχεσιακές βάσεις δεδομένων, αυξάνοντας την αποδοσή τους με μια σειρά νέων χαρακτηριστικών και πλεονεκτημάτων.

Στην παρούσα εργασία επιλέχθηκαν τρεις βάσεις δεδομένων τύπου NoSQL, που ανήκουν σε διαφορετικές κατηγορίες, ώστε να περιγραφούν τα τεχνικά τους χαρακτηριστικά και να γίνει μια ποιοτική σύγκριση μεταξύ τους. Οι βάσεις που επιλέχθηκαν για την ποιοτική μελέτη είναι η Cassandra και η BaseX. Απο την ανάλυση των χαρακτηριστικών τους προκύπτει ότι κάθεμια είναι ιδιαίτερα χρήσιμη ανάλογα με το είδος της εφαρμογής που επιθημεί ο χρήστης να την χρησιμοποιήσει. Εξαρτάται επίσης από το είδος και την ποσότητα των δεδομένων αλλά και τις ανάγκες για την τελική ανάλυση και επεξεργασία των δεδομένων που εισάγονται σε αυτή. Το γενικό συμπέρασμα είναι ότι δεν υπάρχει νικητής και χαμένος σε μια σύγκριση, αλλά αυτή γίνεται για να φανούν με πιο ξεκάθαρο τρόπο τα υπέρ και τα κατά της κάθε βάσης, με απώτερο σκοπό ο χρήστης να επιλέξει το σύστημα που ταιριάζει στις δικές του ανάγκες.

Στο δεύτερο μέρος της εργασίας έγινε μια ποσοτική μελέτη μεταξύ της Cassandra και της Mongo. Οι βάσεις αυτές είναι από τις πιο «διάσημες» στην οικογένεια των NoSQL συστημάτων. Έγιναν διαφόρων ειδών τεστ, που εμπλέκουν διαφορετικά ποσοστά από reads, writes κλπ, ώστε να μπορούμε να έχουμε μια συνολική εικόνα σχετικά με την απόδοση της κάθε βάσης. Αν και είναι γνωστό πως η Cassandra είναι βελτιστοποιημένη στα

writes, ενώ η Mongo στα reads, στην συγκεκριμένη μελέτη, με το συγκεκριμένο σύστημα, η Cassandra στο σύνολο των τεστς, παρουσίασε αρκετά καλύτερη συμπεριφορά από την Mongo όσον αφορά τα χαρακτηριστικά του throughput και του latency. Βέβαια διαφοροποιώντας τις απαιτήσεις και το πως θα στηθεί το σύστημα, τα αποτελέσματα αυτά είναι λογικό πως ενδέχεται να αλλάξουν, κατι το οποίο όμως δεν περιγράφεται στα πλαίσια της εργασίας αυτής.

Βιβλιογραφία

[1] Russom, P. (2011). big data analytics. TDWI Best Practices Report, 4 th Quarter 2011

[2] Luhn, H. P. (1958).A business intelligence system, IBM Journal of Research and Development, 2(4), 314-319

[3] Abadi, D. J. (2009). Data management in the cloud: Limitations and opportunities. IEEE Data Eng. Bull, 32(1), 3-12

[4] Chang, Fay, et al. "Bigtable: A distributed storage system for structured data."ACM Transactions on Computer Systems (TOCS) 26.2 (2008): 4

[5] HBase Databases from web: http://hbase.apache.org/

[6] Lakshman, A., & Malik, P. (2010). Cassandra—A decentralized structured storage system. Operating systems review, 44(2), 35

[7] http://www.slideshare.net/adorepump/voldemort-nosql

[8] Use relational DBMS, N. (2009). Saying good-bye to DBMSs, designing effective interfaces. Communications of the ACM, 52(9)

[9] Padhy, R. P., Patra, M. R., & Satapathy, S. C. (2011). RDBMS to NoSQL: Reviewing Some Next-Generation Non-Relational Database‖. International Journal of Advanced Engineering Science and Technologies, 11(1), 15-30.

[10] Hecht, R., & Jablonski, S. (2011, December). NoSQL evaluation: A use case oriented survey. In Cloud and Service Computing (CSC), 2011 International Conference on (pp. 336-341). IEEE.

[11] Konstantinou, I., Angelou, E., Boumpouka, C., Tsoumakos, D., & Koziris, N. (2011, October). On the elasticity of nosql databases over cloud management platforms. In Proceedings of the 20th ACM international conference on Information and knowledge management (pp. 2385-2388). ACM.

[12] http://www.slideshare.net/skarab/nosql-final

[13]http://www.cems.uwe.ac.uk/~pchatter/2011/dm/readings/rdb_strengths_weaknesses.html

[14] ACID detail from web: http://en.wikipedia.org/wiki/ACID

[15] http://el.wikipedia.org/wiki/ACID

[16] Brewer's CAP Theorem, By Julian Browne on January 11, 2009 web: http://www.julianbrowne.com/article/viewer/brewers-cap-theorem / E.A.Brewer, (2000, Jul) Towards robust distributed systems [http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf]

[17] Han, J., Haihong, E., Le, G., & Du, J. (2011, October). Survey on nosql database. In Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on (pp. 363-366). IEEE

[18] http://martinfowler.com/bliki/NosqlDefinition.html

[19] http://en.wikipedia.org/wiki/NoSQL

[20] R. Cattell, (2010) "Scalable SQL and NoSQL Data Stores," ACM SIGMODRecord, vol. 39

[21]http://readwrite.com/2013/03/25/when-nosql-databases-are-good-for-you

[22] Konstantinou, I., Angelou, E., Boumpouka, C., Tsoumakos, D., & Koziris, N. (2011, October). On the elasticity of nosql databases over cloud management platforms. In Proceedings of the 20th ACM international conference on Information and knowledge management (pp. 2385-2388). ACM

[23] Thantriwatte, T. A. M. C., & Keppetiyagama, C. I. (2011, September). NoSQL query processing system for wireless ad-hoc and sensor networks. InAdvances in ICT for Emerging Regions (ICTer), 2011 International Conference on (pp. 78-82). IEEE

[24] http://blog.sphereinc.com/blog/2012/03/pros-and-cons-of-using-nosql-solutions

[25] http://nosql-database.org/

[26] http://www.monitis.com/blog/2011/05/22/picking-the-right-nosql-database-tool/

[27] http://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases

[28] https://www.mongodb.com/document-databases

[29] https://en.wikipedia.org/wiki/Graph_database

[30] Scofield, Ben (2010-01-14). "NoSQL - Death to Relational Databases(?)". Retrieved 2014-06-26.

[31]http://www.datastax.com/documentation/cassandra/2.0/cassandra/gettingStartedCassandraIntro.html

[32] http://whatis.techtarget.com/definition/Cassandra-Apache-Cassandra

[33] https://academy.datastax.com/demos/brief-introduction-apache-cassandra

[34] http://blog.evanweaver.com/2009/07/06/up-and-running-with-cassandra/

[35] http://www.datastax.com/resources/tutorials/cassandra-overview

[36] http://www.datastax.com/documentation/cassandra/2.0/cassandra/gettingStartedCassandraIntro.html

[37] http://www.ibm.com/developerworks/library/os-apache-cassandra/

[38] https://en.wikipedia.org/wiki/BaseX

[39] http://basex.org/

[40] http://www.networkworld.com/article/2892245/opensource-subnet/basex-free-open-source-xml-wrangling.html

[41] http://basex.org/products/

[42] http://basex.org/products/server/

[43] http://www.rpbourret.com/xml/ProdsNative.htm

[44] Gruen et al. [2007], Visually Exploring and Querying XML with BaseX

[45] http://docs.basex.org/wiki/Transaction_Management

[46] http://basex.org/products/xquery/

[47] http://basex.org/products/gui/

[48]http://www.room4me.com/index.php?option=com_content&view=article&id=8:xmlvsdb&catid=2:technology&Itemid=5

[49] http://xml.coverpages.org/IBM-XML-GC34-2497.pdf

[50] Grolinger et al. Journal of Cloud Computing: Advances, Systems and Applications 2013, 2:22

[51] Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Comm ACM 51(1):107–113. 10.1145/1327452.1327492

[52] http://www.3pillarglobal.com/insights/selection-criteria-for-nosql-database

[53] Moniruzzaman, Hossain, International Journal of Database Theory and Application Vol. 6, No. 4. 2013

[54] http://vschart.com/compare/basex/vs/apache-cassandra

[55] http://db-engines.com/en/system/BaseX%3BCassandra

[56] Sugam Sharma et al., International Journal of Big Data Intelligence 05/2015

[57] https://en.wikipedia.org/wiki/YCSB

[58] Cooper, Brian F et al. "Benchmarking cloud serving systems with YCSB",YahooResearch

[59] http://arunxjacob.blogspot.gr/2011/03/setting-up-ycsb-for-low-latency-data.html

[60] https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads