Ατομική Διπλωματική Εργασία

Μοντελοποίηση δεδομένων και έλεγχος βάσεων δεδομένων στο περιβάλλον του Internet of Things

Γιώργος Μολέσκης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2017

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μοντελοποίηση δεδομένων και έλεγχος βάσεων στο περιβάλλον του Internet of Things

Γιώργος Μολέσκης

Επιβλέπων Καθηγητής

Δρ. Γιώργος Πάλλης

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

Μάιος 2017

Ευχαριστίες

Σε αυτό το μέρος της πτυχιακής μου εργασίας θα ήθελα να ευχαριστήσω όλους όσοι στάθηκαν δίπλα μου κατά τη διάρκεια της εκπόνησης της εργασίας μου καθώς και αυτούς που με την καθοδήγηση τους μπόρεσα να διεκπεραιώσω ίσως το πιο απαιτητικό κομμάτι των σπουδών μου. Καταρχάς θα ήθελα να ευχαριστήσω το Πανεπιστήμιο Κύπρου και το τμήμα Πληροφορικής, τον υπεύθυνο καθηγητή κ. Γιώργος Πάλλη που με τη βοήθεια του επέλεξα ένα ερευνητικό θέμα το οποίο με ενδιαφέρει. Επίσης θα ήθελα να ευχαριστήσω το διδακτορικό φοιτητή του τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου κ. Δημήτρη Τριχινά για τη συνεχή επίβλεψη του.

Δεν θα μπορούσα να παραλείψω τους γονείς μου οι οποίοι με στήριξαν ψυχολογικά και οικονομικά κατά τη διάρκεια εκπόνησης της εργασίας μου καθώς και τα αδέρφια μου για την κατανόηση που έδειξαν. Ευχαριστώ και όλους εσάς που αφιερώνετε μέρος από το χρόνο σας για να διαβάσετε τη πτυχιακή μου εργασία και ελπίζω να τη βρείτε ενδιαφέρουσα .

Περίληψη

Το Διαδίκτυο των πραγμάτων (IoT) έχει γίνει πλέον ένα επίκαιρο θέμα. Αυτή η αυξανόμενη αναγνώριση οφείλεται στο αντίκτυπο που είχε το IoT στην επιχειρηματική ανάλυση και στις δυνατότητες που εξακολουθούν να παραμένουν αναξιοποίητες . Καθημερινά, νέα μηχανήματα, αισθητήρες και συσκευές συνδέονται στο διαδίκτυο και τροφοδοτούν πληροφορίες σε συστήματα δεδομένων. Καθώς οι εταιρίες παίρνουν πρωτοβουλίες και εργάζονται για να αντλήσουν περισσότερες γνώσεις από τον τεράστιο όγκο των δεδομένων, απαιτείται μια νέα προσέγγιση διαχείρισης δεδομένων. Οι παραδοσιακές βάσεις δεδομένων και οι αρχιτεκτονικές ανάλυσης θα είναι πάντα ζωτικής σημασίας για τη συλλογή και ανάλυση δεδομένων. Το IoT απαιτεί συγκεκριμένες δυνατότητες για το χειρισμό ποικίλων δεδομένων λόγω της συνεχούς μετάδοσης δεδομένων από άπειρους αριθμούς πηγών.

Η παρούσα ατομική διπλωματική εργασία ασχολείται με την διαχείριση των διαφορετικών τύπων δεδομένων που υπάρχουν στο περιβάλλον του Internet of Things. Προτείνεται ένα μοντέλο δεδομένων που να καλύπτει τις βασικές ανάγκες και τις ιδιαιτερότητες του Internet of Things το οποίο υλοποιήθηκε σε τέσσερις βάσεις MySql, Cassandra, MongoDb και Coubhbase . Στις προαναφερθείσες βάσεις έγιναν κάποιοι έλεγχοι απόδοσης ώστε να ανακαλύψουμε την πιο ιδανική βάση για τις ανάγκες του Internet of Things καθώς και τα χαρακτηριστικά τους και τις αδυναμίες τους . Υλοποιήθηκε μια διεπαφή προγραμματισμού εφαρμογών ώστε να διαχειρίζεται την σύνδεση, εισαγωγή και εξαγωγή δεδομένων από τις βάσεις και restful service για την διαχείριση των http αιτημάτων. Για κάθε βάση δημιουργήθηκε cluster χρησιμοποιώντας τέσσερα virtual machines . Για την προσομοίωση ενός περιβάλλοντος Internet of Things ώστε να στέλνει δεδομένα στις βάσεις υλοποιήθηκε πρόγραμμα σε Java χρησιμοποιώντας threads και http αιτήματα.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Θεωρητικό Πλαίσιο

Αρχικά , σε αυτό το τμήμα γίνεται μια σύντομη περιγραφή του Internet of things, σημειώνονται τα βασικά πλεονεκτήματα, οι χρήσεις καθώς και οι κύριοι περιορισμοί και ακολουθεί ορισμός της έννοιας μοντέλου δεδομένων . Στη συνέχεια παρουσιάζεται βιβλιογραφική ανασκόπηση . Ακόμα σε αυτό το τμήμα της εργασίας διατυπώνεται ο σκοπός και παραθέτονται τα σενάρια που θα εξεταστούν καθώς και η μεθοδολογία που ακολουθήθηκε .

1.1.α Internet of Things

Ο όρος Internet of Things επινοήθηκε το 1999 από τον Kevin Ashton για να περιγράψει ένα σύστημα όπου το Internet είναι συνδεδεμένο με τον φυσικό κόσμο μέσω αισθητήρων [1] . Ο ορισμός έχει εξελιχθεί με το πέρασμα του χρόνου και είναι πλέον πιο περιεκτικός. Καλύπτει ένα ευρύ φάσμα εφαρμογών, όπως η υγειονομική περίθαλψη, οι επιχειρήσεις κοινής ωφέλειας, οι επιχειρήσεις μεταφορών κ. α. . Πιο συγκεκριμένα το Internet of Things είναι ένα δίκτυο έξυπνων αντικειμένων συνδεδεμένων στο διαδίκτυο που μπορούν να συλλέγουν και να ανταλλάσσουν δεδομένα χρησιμοποιώντας ενσωματωμένους αισθητήρες σχηματίζοντας έτσι το Internet of Things.

Στόχος είναι τα έξυπνα αντικείμενα που αποτελούνται από διάφορους αισθητήρες να συνδέονται αρμονικά με το περιβάλλον γύρω μας, ώστε να είναι δυνατή η συλλογή πληροφοριών χωρίς την ανθρώπινη παρέμβαση . Το Internet of Things στοχεύει στην αφαίρεση της έννοιας του τεχνολογικού κατασκευάσματος από τη συνείδηση του χρήστη. Αυτό είναι πραγματοποιήσιμο μόνο όταν γίνει πλήρως κατανοητή η κατάσταση των χρηστών αλλά και των συσκευών τους και εφόσον αναπτυχθούν κατάλληλες αρχιτεκτονικές λογισμικού για την επεξεργασία και τη μεταφορά πληροφοριών [1]. Επίσης η ανάπτυξη εργαλείων ανάλυσης με στόχο την αυτόνομη και έξυπνη συμπεριφορά τους αποτελεί παράγοντα που δύναται να συμβάλει στην επίτευξη έξυπνης συνδεσιμότητας .

Αναμφισβήτητα, το κύριο πλεονέκτημα της ιδέας του IoT είναι το μεγάλο αντίκτυπο που θα έχει στις διάφορες πτυχές της καθημερινής ζωής και της συμπεριφοράς των δυνητικών χρηστών. Από την άποψη ενός ιδιώτη χρήστη, τα πιο προφανή αποτελέσματα της εισαγωγής του IoT θα είναι ορατά τόσο στον εργασιακό όσο και στον οικιακό χώρο. Σε αυτό το πλαίσιο, τα νοικοκυριά, η υποβοηθούμενη

διαβίωση, η ηλεκτρονική υγεία, η βελτιωμένη μάθηση είναι μόνο μερικά παραδείγματα πιθανών σεναρίων εφαρμογής στα οποία το νέο πρότυπο θα διαδραματίσει ηγετικό ρόλο στο εγγύς μέλλον [2] . Ομοίως, από την οπτική γωνία των επιχειρηματικών χρηστών, οι πιο εμφανείς συνέπειες θα είναι αισθητές σε τομείς όπως ο αυτοματισμός και η βιομηχανική παραγωγή, ο εφοδιασμός, η διαχείριση των επιχειρήσεων / διαδικασιών, η ευφυής μεταφορά ατόμων και αγαθών.

1.1.β Προκλήσεις του Internet of things

Υπάρχουν αρκετοί περιορισμοί και προκλήσεις οι οποίοι προκύπτουν από τη χρήση του Internet of Things και είναι επιτακτική η μελέτη και βελτιστοποίηση πολλών ζητημάτων που σχετίζονται με την προστασία της ιδιωτικής ζωής (privacy), την ασφάλεια (security), την επεκτασιμότητα (scalability), την καθυστέρηση (latency), το εύρος ζώνης (bandwidth), τη διαθεσιμότητα (availability) και τον έλεγχο της ανθεκτικότητας (durability control) [3].

Είναι αναγκαίο να αντιμετωπιστούν πολλά ζητήματα τεχνολογικά και κοινωνικά ενώ οι προτεινόμενες λύσεις πρέπει να εστιάσουν στην αποδοτικότητα των πόρων. Το Internet of Things τροφοδοτεί τεράστια ποσά δεδομένων με συνεχείς ροές (streams) δεδομένων από συσκευές που είναι διάσπαρτες σε πολλαπλές τοποθεσίες όπου σημαντικό ρόλο έχουν τα δεδομένα πραγματικού χρόνου (real-time)[5].

Οι λύσεις για τη διαχείριση και τη χρήση του τεράστιου όγκου δεδομένων που παράγονται από αυτά τα αντικείμενα δεν έχουν ακόμη ωριμάσει. Επομένως, το Internet of things και ο όρος Big Data αναφέρονται συχνά ως δύο όψεις του ίδιου νομίσματος [5]. Ωστόσο, η συλλογή και ανάλυση μεγάλων όγκων πρωτογενών δεδομένων παρακολούθησης είναι τόσο ακριβή όσο και χρονοβόρα και μπορεί να συνεπάγεται με

δαπανηρές λειτουργίες, όπως ανάλυση μεγάλων αρχείων καταγραφής ή εκτέλεση πολύπλοκων ερωτημάτων. Τα δεδομένα όχι μόνο αυξάνονται, αλλά και διαφοροποιούνται κάνοντας απαραίτητη τη χρήση εργαλείων και υποδομών που απαιτούνται για την αξιοποίηση των μη παραδοσιακών μορφών δεδομένων. Με τη δημιουργία του Internet of Things παρουσιάζεται ένα νέο σύνολο προκλήσεων στα συστήματα διαχείρισης βάσεων όπως η λήψη μεγάλων όγκων δεδομένων σε πραγματικό χρόνο ή σχεδόν σε πραγματικό χρόνο, η επεξεργασία γεγονότων και η αντιμετώπιση πολύ μεγαλύτερων όγκων δεδομένων από ό, τι συχνά συναντάται.

Υπάρχουν αρκετά ερευνητικά προγράμματα στο περιβάλλον του Internet of Things, παρατηρείται όμως έλλειψη υιοθέτησης κατάλληλων πρωτοτύπων. Είναι απαραίτητο , καθώς ωριμάζει η τεχνολογία, να διασφαλιστεί η λειτουργικότητα μεταξύ συσκευών και εφαρμογών που προέρχονται από διαφορετικά συστήματα [9]. Τα πρωτόκολλα πρέπει να είναι σχεδιασμένα ώστε να υποστηρίζουν ένα ευρύ φάσμα εφαρμογών και να αντιμετωπίζουν τις απαιτήσεις και προκλήσεις του περιβάλλοντος . Έχουν ξεκινήσει κάποιες προσπάθειες τυποποίησης και δημιουργίας πρωτοκόλλων στον τομέα των αισθητήρων, επικοινωνίας και μεθόδων εξέτασης [9]. Παρόλα αυτά υπάρχει σαφής έλλειψη δραστηριοτήτων τυποποίησης που σχετίζονται με μοντέλα και τύπους δεδομένων που μπορούν να χρησιμοποιηθούν στον περιβάλλον του Internet of Things.

1.2 Σκοπός Εργασίας

Σκοπός της παρούσας εργασίας είναι η δημιουργία ενός μοντέλου δεδομένων για τις ανάγκες του Internet of things το οποίο θα αποτελέσει εργαλείο για τον εντοπισμό της ιδανικής βάσης δεδομένων για το Internet of things. Για τους σκοπούς της εργασίας δημιουργήθηκε ένα μοντέλο δεδομένων που να πληρεί τις ανάγκες του Internet of Things. Με βάση αυτό το μοντέλο δημιουργήθηκαν τα σχήματα των βάσεων δεδομένων (database schema) MySql, η Cassandra, η MongoDb και η CouchBase οι οποίες θα εξεταστούν ως προς την απόδοση τους. Για να είναι δυνατός ο έλεγχος των βάσεων υλοποιήθηκε και μια διεπαφή προγραμματισμού εφαρμογών (API) η οποία ελέγχει την εισαγωγή και εξαγωγή δεδομένων από τις πιο πάνω βάσεις όπως και το resful service για την διαχείριση των αιτημάτων. Η συνεισφορά της διπλωματικής εργασίας είναι ένα προτεινόμενο μοντέλο δεδομένων για την διαχείριση των πολλών διαφορετικών τύπων δεδομένων που υπάρχουν στο Internet of Things και η διεπαφή προγραμματισμού εφαρμογών (API) που μπορεί να διαχειρίζεται αιτήματα HTTP και να αποθηκεύει τα δεδομένα σε μια από τις τέσσερις βάσεις δεδομένων που είναι υλοποιημένες, έχοντας την δυνατότητα να υλοποιηθούν οι διεπαφές για οποιαδήποτε βάση δεδομένων .

1.3 Σενάρια που θα εξεταστούν

Πιο κάτω παραθέτονται τα σενάρια που θα εξεταστούν. Κάθε σενάριο αφορά και τις τέσσερις προαναφερθείσες βάσεις (mysql, cassandra , mongodb και couchbase) και θα αναλυθεί και θα παρουσιαστεί σε μεταγενέστερο κεφάλαιο. Οι τέσσερις βάσεις θα

χρησιμοποιούν το ίδιο ευέλικτο μοντέλο δεδομένων προσαρμοσμένο με τους κανόνες κάθε βάσης ώστε να είναι δυνατή η σύγκρισή τους.

1. Αριθμός αιτημάτων που ικανοποιούνται το δευτερόλεπτο .
2. Χρόνος που χρειάζεται για εγγραφή σε σχέση με τον αριθμό των αισθητήρων που στέλνουν δεδομένα .
3. Χρόνος που χρειάζεται από την στιγμή που στέλνουμε τα δεδομένα μέχρι να είναι έτοιμα για παρουσίαση .
4. Χρόνος που χρειάζεται για να διαβαστεί μια εγγραφή όταν οι αιτήσεις είναι 50% για ανάγνωση και 50% εγγραφή .
5. Χρόνος που χρειάζεται για να αποθηκευτεί μια εγγραφή όταν οι αιτήσεις είναι 50% για ανάγνωση και 50% εγγραφή .
6. Αριθμός των ανοιχτών συνδέσεων που μπορούν να διαχειριστούν οι βάσεις .
7. Χρόνος για εισαγωγή των δεδομένων στη βάση αυξάνοντας χρησιμοποιώντας τυχαία περίοδο αποστολής .
8. Ανάγνωση ενός εκατομμυρίου εγγραφών από την βάση όταν οι αποθηκευμένες εγγραφές είναι ένα εκατομμύριο, δέκα εκατομμύρια και είκοσι εκατομμύρια .
9. Ο χρόνος ανάκτησης, 10000 εγγραφών, 100000 και 1000000 .
10. Έλεγχος χρόνου δημιουργίας σύνδεσης με την βάση .

Παράλληλα θα παρακολουθείται (monitoring) η κατάσταση των μηχανών ελέγχοντας κάθε δευτερόλεπτο την χρήση του επεξεργαστή (CPU) και τη μνήμη τυχαίας προσπέλασης (RAM) για την μηχανή που παράγει δεδομένα και των clusters για κάθε βάση δεδομένων . Το κύριο ερώτημα το οποίο στοχεύουμε να απαντηθεί είναι ποιά βάση δεδομένων μπορεί να μας δώσει τα καλύτερα αποτελέσματα όταν προσομοιώνουμε τα πιο πάνω σενάρια . Δηλαδή ποιά βάση μπορεί να διαχειριστεί πολλούς αισθητήρες που στέλνουν δεδομένα στην βάση έχοντας μικρό χρόνο εγγραφής και ανάγνωσης, όπως και μικρό χρόνο καθυστέρησης από την αποστολή των δεδομένων μέχρι την δυνατότητα ανάκτησης τους .

1.4 Μεθοδολογία

Σε αυτό το μέρος της διπλωματικής εργασίας παρατίθενται τα βήματα που ακολουθήθηκαν για τη διεξαγωγή του πειραματικού μέρους . Αρχικά έγινε έρευνα στο περιβάλλον του Internet of things ώστε να ανακαλυφθούν οι ανάγκες του και έπειτα σχεδιασμός “ high level solution ”, δηλαδή δομήθηκε η αρχιτεκτονική του συστήματος και καθορίστηκε ένα μοντέλο δεδομένων το οποίο χαρακτηρίζεται από λειτουργικότητα και αποτελεσματικότητα για όλους τους τύπους δεδομένων . Στη συνέχεια επιλέχθηκαν οι βάσεις δεδομένων MySql, Cassandra, MongoDB, Couchbase . Ακολούθησε η υλοποίηση του κώδικα κατά την οποία δημιουργήθηκε API για εισαγωγή δεδομένων για κάθε βάση , API για ανάκτηση δεδομένων από τις βάσεις . Τέλος εκτελέστηκε δοκιμή και σύγκριση των βάσεων αλλά και έλεγχος λειτουργικότητας .

Κεφάλαιο 2

Περιγραφή προβλήματος και ανασκόπηση βιβλιογραφίας

2.1 Internet of Things

2.1.1 Περιγραφή

Οι πρόσφατες εξελίξεις στη μικροηλεκτρονική, τις τηλεπικοινωνίες και την εξόρυξη δεδομένων έχουν οδηγήσει σε μια αυξανόμενη υιοθέτηση έξυπνων συσκευών, οι οποίες επηρεάζουν τον τρόπο με τον οποίο ζούμε και εργαζόμαστε . Αυτές οι συσκευές καταγράφουν και ανταλλάσσουν συνεχείς ροές δεδομένων με άλλες συσκευές και έχουν δυνατότητα σύνδεσης σε δίκτυο, σχηματίζοντας, όπως είναι γνωστό, το

Διαδίκτυο των πραγμάτων (IoT) . Οι συσκευές IoT διαφέρουν από τους αισθητήρες καθώς διαθέτουν δυνατότητες επεξεργασίας για την παραγωγή αναλυτικών στοιχείων από ακατέργαστα δεδομένα παρακολούθησης .

Τα δίκτυα αισθητήρων μεγάλης κλίμακας απαιτούν αυστηρές τεχνικές διαχείρισης και διάδοσης δεδομένων . Η ασύρματη επικοινωνία η οποία προκαλεί αυξημένη κατανάλωση, καθώς και η ασυνέχεια που μπορεί να προκληθεί στην τροφοδοσία των αισθητήρων (λόγω περιορισμένου εύρους ζώνης) αποτελούν περιοριστικούς παράγοντες για τη μετάδοση δεδομένων από τους αισθητήρες στο σταθμό της βάσης . Το σύστημα πρέπει να είναι ευέλικτο ώστε να είναι δυνατή η αντιμετώπιση προβλημάτων που προκύπτουν (όνομα και διεύθυνση αντικειμένου) από την καθημερινή σύνδεση νέων αντικειμένων, καθώς και προβλήματα επικοινωνίας και μεταφοράς δεδομένων μέσα στο δίκτυο.

Επιπλέον τίθεται το θέμα της ασφάλειας αντικειμένων, αφού το IoT αποτελείται από ένα μεγάλο αριθμό που διαδίδουν ευαίσθητα δεδομένα όπως η τοποθεσία . Είναι απαραίτητο να αποτραπεί η πρόσβαση σε άτομα ή οργανισμούς που μπορεί να προκαλέσουν δυσλειτουργία ,ακούσια ή εκούσια, ή να αλλάξουν την λειτουργία τους [4] .Τα δεδομένα στέλνονται μέσω δικτύου το οποίο πρέπει να είναι ικανό να διαχειριστεί τις μεγάλες ποσότητες δεδομένων χωρίς την απώλεια δεδομένων και να εξασφαλίσει την ασφάλεια χωρίς την παρέμβαση και παρακολούθηση από εξωτερικούς παράγοντες .

2.1.2 Διαχείριση Δεδομένων

Η διαχείριση δεδομένων είναι μια ευρεία έννοια που αναφέρεται στις αρχιτεκτονικές, πρακτικές και διαδικασίες για την ορθή διαχείριση των αναγκών ενός κύκλου ζωής ενός συγκεκριμένου συστήματος . Στο πλαίσιο του Internet Of Things, η διαχείριση δεδομένων θα πρέπει να ενεργεί ως στρώμα μεταξύ των αντικειμένων και των συσκευών που παράγουν τα δεδομένα και των εφαρμογών που έχουν πρόσβαση

στα δεδομένα για σκοπούς ανάλυσης των δεδομένων και προσφορά κάποιων υπηρεσιών . Οι ίδιες οι συσκευές μπορούν να διευθετηθούν σε υποσυστήματα ή υποπεριοχές με αυτόνομη διακυβέρνηση και εσωτερική ιεραρχική διαχείριση . Η λειτουργικότητα και τα δεδομένα που παρέχονται από αυτά τα υποσυστήματα πρέπει να διατίθενται στο δίκτυο IoT, ανάλογα με το επίπεδο προστασίας της ιδιωτικής ζωής που επιθυμούν οι ιδιοκτήτες του υποσυστήματος .

Το Internet of Things τροφοδοτεί τεράστια ποσά δεδομένων με συνεχείς ροές (streams) δεδομένων από συσκευές που είναι διάσπαρτες σε πολλαπλές τοποθεσίες όπου σημαντικό ρόλο έχουν τα δεδομένα πραγματικού χρόνου (real-time) [5] . Όσο πιο γρήγορα έχουμε πρόσβαση στα δεδομένα τόσο μεγαλύτερη είναι η αξία τους . Για παράδειγμα οι εφαρμογές όπως είναι η έξυπνη πόλη και η διαχείριση της κίνησης στους δρόμους δεν θα μπορούσαν να υφίστανται χωρίς τα δεδομένα πραγματικού χρόνου . Τα δεδομένα αυτά είναι δυνατόν να προέλθουν από καλά σχεδιασμένες υποδομές ειδικών εφαρμογών ή από συλλογή από το κοινό . Ωστόσο, είναι δύσκολο να συγκεντρωθούν ή να κοινοποιηθούν δεδομένα που ανήκουν σε άτομα που έχουν διαφορετικές συνήθειες, γνώσεις, αντιλήψεις, εισόδημα, οφέλη από την κοινωνία, κοινωνικές ευθύνες, κ. α. . Η πλατφόρμα που επιτρέπει τέτοιου είδους δραστηριότητες συλλογής και ανταλλαγής πρέπει να έχει μηχανισμούς με τους οποίους θα παρακινεί τους ιδιοκτήτες των δεδομένων να τα μοιραστούν [6] .

Τα δεδομένα όχι μόνο αυξάνονται, αλλά και διαφοροποιούνται κάνοντας απαραίτητη τη χρήση εργαλείων και υποδομών που απαιτούνται για την αξιοποίηση των μη παραδοσιακών μορφών δεδομένων . Τα δεδομένα μπορεί να είναι μονοδιάστατα, πολυδιάστατα,να έχουν μορφή απλού κειμένου, τοποθεσίας (μήκος, πλάτος) ή spatio-temporal . Αρκετή ιδιαιτερότητα παρουσιάζουν τα spatio-temporal δεδομένα τα οποία υποδηλώνουν την τοποθεσία ενός αντικειμένου στον χώρο και χρόνο . Τα δεδομένα ενημερώνονται συνεχώς με αποτέλεσμα να καταγράφεται οποιαδήποτε αλλαγή στην γεωμετρία (geometry) κατά την διάρκειά του χρόνου [7] .

Τα spatio-temporal δεδομένα παριστάνονται με ένα σημείο (point) που είναι η τοποθεσία του αντικειμένου, μια γραμμή (line) που περιγράφει την κίνηση ή τη σύνδεση του στον χώρο και τη περιοχή (region) που σχετίζεται το αντικείμενο [7].

2.1.3 Βάσεις Δεδομένων

Η δομή της βάσης δεδομένων βασίζεται σε ένα μοντέλο δεδομένων το οποίο ορίζει τον τρόπο που περιγράφονται τα δεδομένα, οι σχέσεις τους, η σημασία τους και οι περιορισμοί πάνω στα δεδομένα αυτά . Τα μοντέλα δεδομένων καθορίζουν πως είναι η λογική δομή μιας βάσης δεδομένων . Στη βάση υπάρχουν οι θεμελιώδεις οντότητες που εισάγουν την έννοια της αφαιρετικότητας σε μια βάση δεδομένων . Καθορίζουν πως τα δεδομένα είναι συνδεδεμένα μεταξύ τους καθώς και τον τρόπο επεξεργασίας και αποθήκευσης τους στο σύστημα . Στο περιβάλλον του IoT είναι επιτακτική η ανάγκη για ένα μοντέλο που να είναι ευέλικτο ώστε να υποστηρίζει την διαφορετικότητα των δεδομένων . Οι παραδοσιακές λύσεις διαχείρισης βάσεων δεδομένων δεν είναι ικανές να καλύψουν τις ανάγκες του Internet of Things .

Στα παραδοσιακά συστήματα διαχείρισης δεδομένων τα δεδομένα συλλέγονται από ορισμένο αριθμό σημείων που είναι γνωστά στο σύστημα και αποθηκεύονται σε συγκεκριμένη μορφή, κανόνες και συσχετίσεις . Τα ερωτήματα (queries) χρησιμοποιούνται για εισαγωγή δεδομένων, ανάκτηση και ενημέρωση τους. Μηχανισμοί διαχείρισης συναλλαγών εγγυώνται την ακεραιότητα των δεδομένων και την επιβολή ιδιοτήτων ACID (Atomicity, Consistency, Isolation, Durability) [8] . Ατομικότητα απαιτεί η κάθε συναλλαγή να είναι όλα ή τίποτα δηλαδή εάν ένα κομμάτι της συναλλαγής αποτύχει τότε θα αποτύχει όλη η συναλλαγή χωρίς να αλλάξει την κατάσταση της βάσης. Συνέπεια εξασφαλίζει ότι η βάση θα βρίσκεται πάντα σε σταθερή κατάσταση . Η απομόνωση εγγυάται ότι η ταυτόχρονη εκτέλεση συναλλαγών έχει το ίδιο αποτέλεσμα εάν οι συναλλαγές εκτελούνται η μια μετά την άλλη . Η αντοχή είναι η ιδιότητα η οποία επιτρέπει , όταν εκτελεστεί μια συναλλαγή , να παραμείνει στην βάση χωρίς να επηρεάζεται απώλεια ισχύος ή οποιαδήποτε σφάλματα.

Στο περιβάλλον του Internet of Things υπάρχουν αμέτρητες πηγές δεδομένων που συνεχώς μπορεί να αυξάνονται. Ταυτόχρονα, το Διαδίκτυο επιβάλλει λιγότερους περιορισμούς ποιότητας και ακεραιότητας δεδομένων . Για παράδειγμα, μια εφαρμογή μπορεί να ανεχτεί την απώλεια δεδομένων για μερικά λεπτά και να εξακολουθεί να είναι σε θέση να παρακολουθεί τις δυνατότητες λειτουργίας . Αν και οι αισθητήρες IoT παράγουν δεδομένα γρήγορα,το γεγονός αυτό δεν συνεπάγεται τις ίδιες συναλλαγές που συναντά κανείς στις παραδοσιακές επιχειρηματικές εφαρμογές. Αυτό μειώνει την ανάγκη για συναλλαγές ACID (Atomicity, Consistency, Isolation, Durability) . Δεν υπάρχει αυστηρό σχεσιακό μοντέλο και προτιμούνται πιο αδόμητες και ευέλικτες μορφές που προσαρμόζονται στην διαφορετικότητα των δεδομένων . Οι βάσεις δεδομένων IoT πρέπει να είναι γραμμικώς κλιμακούμενες (scalable) και διανεμημένες (destributed) και τόσο ευέλικτες όσο απαιτείται από την εφαρμογή [8] .

2.2 Παρόμοια συστήματα

2.2.1 MARSA: A Marketplace for Realtime Human Sensing Data

Marsa είναι μια δυναμική πλατφόρμα βασισμένη στο cloud η οποία διαθέτει ,σε σχεδόν πραγματικό χρόνο, δεδομένα τα οποία οι ενδιαφερόμενοι μπορούν να πουλήσουν ή να αγοράσουν . Είναι σχεδιασμένη για περιβάλλοντα όπου οι εγκαταστάσεις συλλογής πληροφορίας δεν είναι αρκετά ανεπτυγμένες, ώστε να εξασφαλίσουν δεδομένα σε πραγματικό χρόνο, δημιουργώντας την ανάγκη άντλησης αυτών των δεδομένων από μεσάζοντες . Στο σχήμα 2.1.1 φαίνεται ο τρόπος οργάνωσης και λειτουργίας του Marsa . Άτομα ή εταιρείες, που έχουν πρόσβαση σε δεδομένα πραγματικού χρόνου,μπορούν να πουλήσουν τα εν λόγω δεδομένα μέσω της

πλατφόρμας . Παράδειγμα παρόμοιας συναλλαγής (στην οποία παρέχετε κίνητρο στο χρήστη ώστε να προσφέρει τα δεδομένα του) αποτελεί η περίπτωση στην οποία ο χρήστης ανταμείβεται, για την παροχή δεδομένων, με ενημέρωση από την εταιρεία με την οποία συνεργάζεται, η οποία αφορά την τροχαία κίνηση .

Δίνοντας την δυνατότητα στους χρήστες να πουλήσουν οποιοδήποτε τύπο δεδομένων σχεδιάστηκε ένα μοντέλο που να μπορεί να υποστηρίξει αυτή την ιδέα . Χρησιμοποιήθηκε entity-relationship μοντέλο όπου εκτός από τις βασικές οντότητες προστέθηκαν ιδιότητες χρόνου (data rate, latency, and time series) και τύποι δεδομένων (type of data) με σκοπό την επιτυχή διαχείριση των συνεχών ροών και τύπων δεδομένων που μπορεί να υπάρξουν. Η βάση που χρησιμοποιήθηκε είναι η MySql .

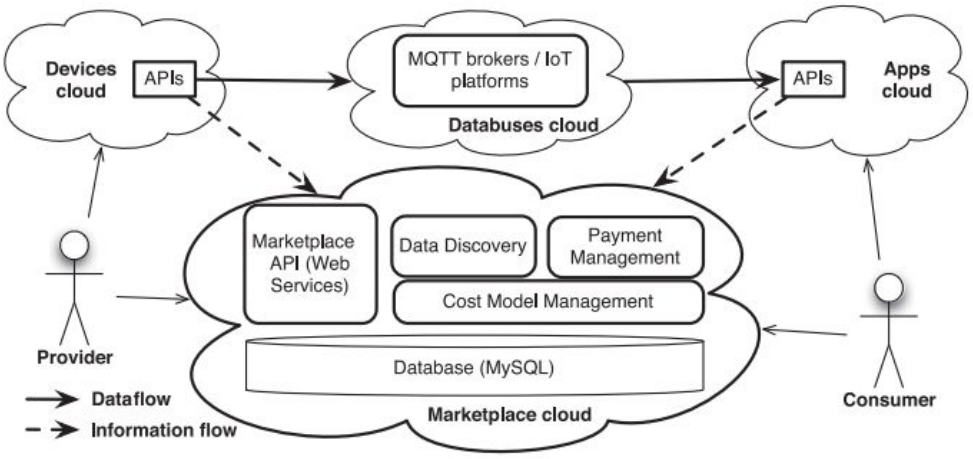


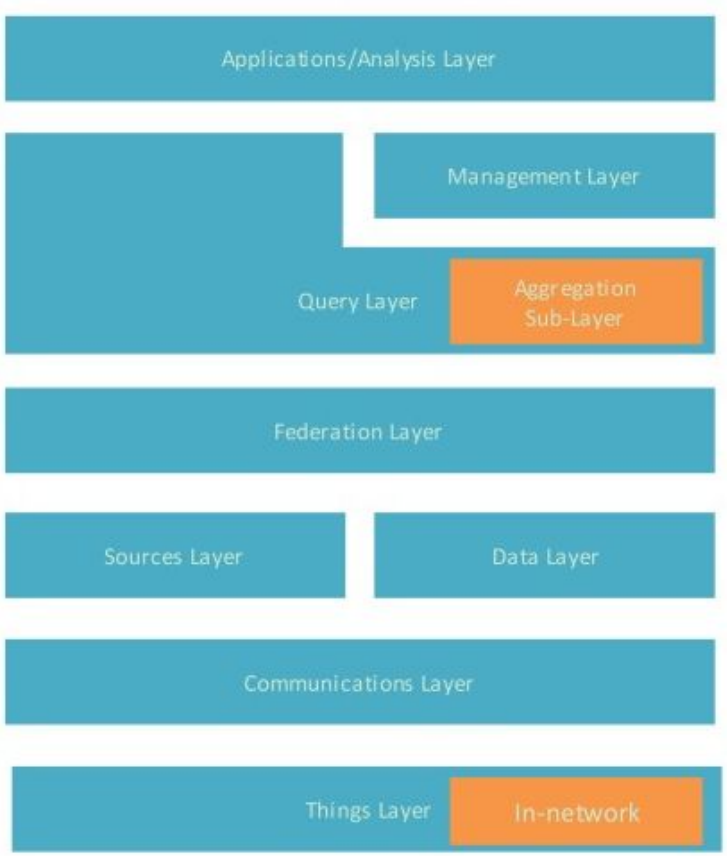
Fig. 4. MARSA prototype.

Σχήμα 2.1.1 Πρωτότυπο MARSA , τρόπος οργάνωσης και λειτουργίας. [6]

2.2.2 Design Primitives and Solution

Οι συγγραφείς στο άρθρο τους Design Primitives and Solution προτείνουν ένα framework για διαχείριση των δεδομένων στο Internet of Things . Το προτεινόμενο framework αποτελείται από έξι επίπεδα όπως φαίνεται και στο σχήμα 2.2.2.1 . Στο πιο χαμηλό επίπεδο κατατάσσονται τα αντικείμενα (Things) και αποτελείται από αισθητήρες ή έξυπνα αντικείμενα . Ακολουθεί η επικοινωνία (Communication Layer) που περιλαμβάνει την μεταφορά αιτημάτων, δεδομένων και αποτελεσμάτων . Στο δεύτερο επίπεδο βρίσκονται επίσης τα δεδομένα και οι πηγές (Data / Sources Layer) τα οποία συνδέονται με την κατηγοριοποίηση, την αποθήκευση και τη δημιουργία ευρετηρίων στα δεδομένα καθώς και με την απάντηση σε ερωτήματα (query) . Το Federation layer χρησιμοποιώντας μεταδεδομένα που βρίσκονται στο Data / Sources Layer , προεπεξεργάζεται τα δεδομένα ώστε να μπορεί να υποστηρίξει την μετάδοση δεδομένων σε πραγματικό χρόνο, συνδέει τους διάφορους τύπους δεδομένων που μπορεί να υπάρχουν ώστε να απαντήσει ένα συγκεκριμένο ερώτημα . Το Query Layer διαχειρίζεται τις λεπτομέρειες στην επεξεργασία και βελτιστοποίηση των ερωτημάτων σε συνεργασία με το Federation Layer .

Στο Data Layer προτείνεται μια υβριδική λύση για αποθήκευση των δεδομένων . Τα προσωρινά δεδομένα ή πραγματικού χρόνου αποθηκεύονται κοντά στα αντικείμενα που τα παράγουν και τα υπόλοιπα δεδομένα που πρέπει να αποθηκευτούν μακροπρόθεσμα για ανάλυση και κατηγοριοποίηση αποθηκεύονται σε διαφορετικές εγκαταστάσεις . Με αυτό τον τρόπο αυξάνεται το κόστος αλλά παράλληλα βελτιώνεται η διαθεσιμότητα των δεδομένων και η ταχύτητα με την οποία μπορούμε να τα ανακτήσουμε . Δεν προτείνεται συγκεκριμένο μοντέλο δεδομένων στο άρθρο όμως αναφέρεται ότι χρειάζεται μια πιο ευέλικτη δομή δεδομένων για τις συγκεκριμένες ανάγκες .



Σχήμα 2.2.2.1 Αρχιτεκτονική συστήματος [8]

2.2.3 Benchmarking In-Network Sensor Query Processing

Το Bisque που είναι ένα γενικό εργαλείο ελέγχου επίδοσης, που αποκαλύπτει τα κύρια χαρακτηριστικά των συστημάτων [10] . Αποτελείται από 3 επίπεδα : Mac Layer, Royting Layer και Query Layer . Κάποιες από τις λειτουργίες του είναι η δημιουργία φόρτου εργασίας (workload generator) από ερωτήματα (queries), μέτρηση μετρικών επίδοσης και αύξηση αριθμού αισθητήρων .

Το μοντέλο δεδομένων που χρησιμοποιείται αποτελείται από μία οντότητα με όνομα sensors . Περιέχει χαρακτηριστικά όπως είναι η ταυτότητα του αισθητήρα, η ημερομηνία και η τοποθεσία . Είναι σχεδιασμένο ώστε να υποστηρίζει απλούς τύπους δεδομένων όπως θερμοκρασία και φως . Για τον έλεγχο της επίδοσης δημιουργήθηκε workload generator και έγινε επιλογή κάποιων ερωτημάτων τα οποία παρουσιάζονται πιο συχνά με βάση το μοντέλο δεδομένων . Οι μετρικές στις οποίες εφαρμόζεται έλεγχος είναι η κατανάλωση ενέργειας του κόμβου που επεξεργάζεται κάποιο ερώτημα, ο χρόνος απάντησης σε κάποιο ερώτημα και ο αριθμός των σφαλμάτων .

Κεφάλαιο 3

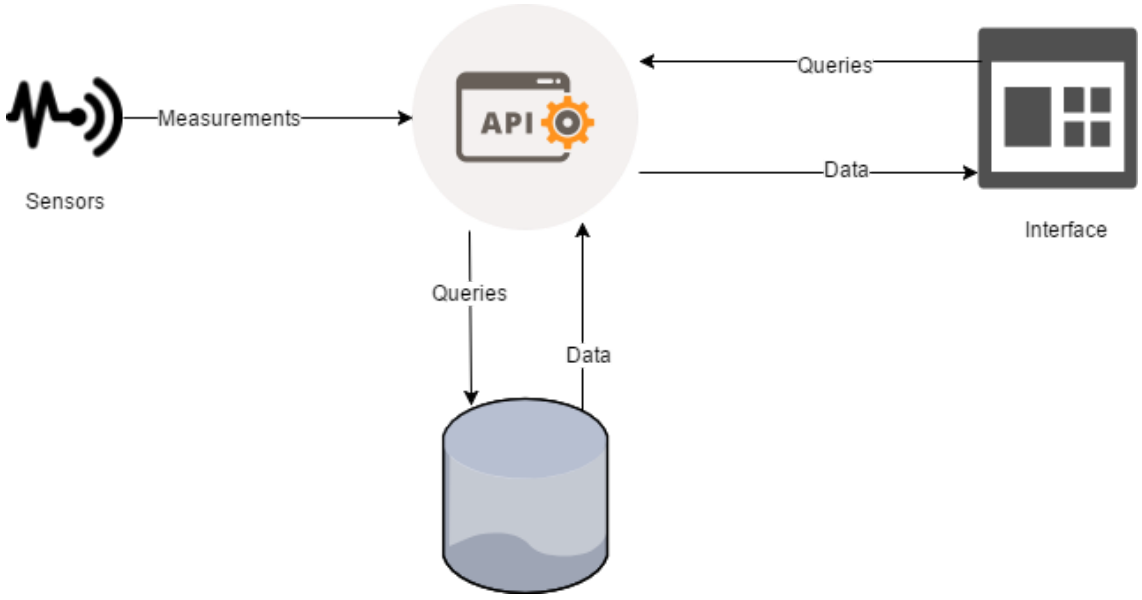
Περιγραφή Υλοποίησης

3.1 Γενική περιγραφή

Για την υλοποίηση της διπλωματικής εργασίας και τον έλεγχο των ερευνητικών ερωτημάτων αρχικά καθορίστηκε η αρχιτεκτονική του συστήματος και έπειτα σχεδιάστηκε ένα μοντέλο δεδομένων για τις ανάγκες του Internet of Things . Δημιουργήθηκε διεπαφή προγραμματισμού εφαρμογών (API) σε γλώσσα προγραμματισμού Java ώστε να γίνεται η εισαγωγή και εξαγωγή δεδομένων από τις βάσεις και κτίστηκε RESTful api ώστε να γίνονται δεκτά τα αιτήματα και να εξάγονται οι πληροφορίες . Επιπλέον για να είναι δυνατό να γίνει ο έλεγχος των βάσεων δημιουργήθηκε πρόγραμμα σε Java ώστε να παράγει φόρτο εργασίας . Σε αυτό το κεφάλαιο θα περιγραφεί η υλοποίηση για τα πιο πάνω .

3.2 Αρχιτεκτονική Συστήματος

Στο περιβάλλον του Internet of Things ο κύκλος ζωής των δεδομένων ξεκινά στον αισθητήρα που παράγει κάποια μέτρηση την οποία φιλτράρει και επεξεργάζεται και ακολούθως την στέλνει για αποθήκευση και αρχειοθέτηση . Μόλις αποθηκευτεί η μέτρηση είναι διαθέσιμη ,μετά από αίτηση μέσω των ερωτημάτων (queries) . Έτσι στην αρχιτεκτονική του συστήματος στο πιο χαμηλό επίπεδο βρίσκονται οι αισθητήρες, οι οποίοι στέλνουν τα δεδομένα σε μορφή Json (που περιέχει το id, timestamp) και την τιμή της μέτρησης . Στον server έχουμε ένα API (application programming interface) που λαμβάνει τα δεδομένα μέσω POST αιτημάτων και τα αποθηκεύει στην βάση δεδομένων . Επιπλέον μέσω GET αιτήματα γίνεται ανάκτηση των δεδομένων, το API θα αναλύσει το αίτημα και θα απαντήσει . Στο σχήμα 3.2.1 παρουσιάζεται η αρχιτεκτονική του συστήματος .

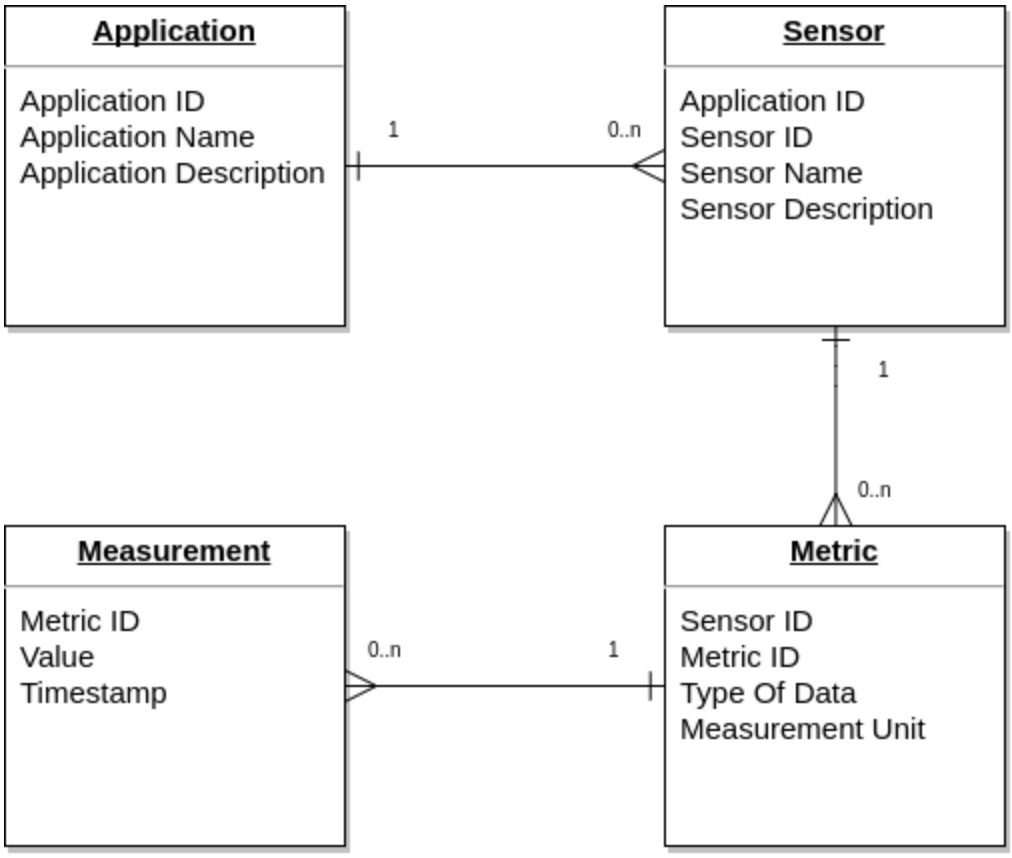


Σχήμα 3.2.1 Σχηματική απεικόνιση αρχιτεκτονικής συστήματος .

3.3 Περιγραφή Μοντέλου δεδομένων

Το μοντέλο που θα χρησιμοποιηθεί αποτελείται από από τέσσερις οντότητες που είναι οι εφαρμογές, οι αισθητήρες, οι μετρικές και οι μετρήσεις και παριστάνεται σχηματικά στο σχήμα 3.3.1 . Κάθε εφαρμογή έχει αμέτρητους αισθητήρες, κάθε αισθητήρας μπορεί να έχει αμέτρητες μετρικές και κάθε κάθε μετρική μπορεί να έχει αμέτρητες μετρήσεις όπως φαίνεται και στο πιο κάτω σχήμα .

Για κάθε εφαρμογή αποθηκεύουμε όνομα, περιγραφή και ταυτότητα. Κατά την δημιουργία της εφαρμογής το πρόγραμμα επιστρέφει μια ταυτότητα τύπου UUID (Universally unique identifier) . Το UUID είναι ένας 128-bit αριθμός με πιθανότητα να υπάρχει αντίγραφο σχεδόν 0 έτσι είμαστε σχεδόν σίγουροι ότι η ταυτότητα μας είναι μοναδική . Αποτελείται από 32 δεκαεξαδικούς αριθμούς, συνολικά 36 χαρακτήρες για παράδειγμα 123e4567-e89b-12d3-a456-426655440000 . Σε αντίθεση με την ταυτότητα της εφαρμογής, το όνομα και η περιγραφή είναι δυνατό να αλλάξουν . Στον αισθητήρα υπάρχει η ταυτότητα της εφαρμογής στην οποία ανήκει ο αισθητήρας όπως και η ταυτότητα του αισθητήρα . Οι ταυτότητες και στις δύο περιπτώσεις δημιουργούνται με τον ίδιο τρόπο . Επιπλέον σε κάθε αισθητήρα υπάρχει το όνομα και η περιγραφή του . Για κάθε μετρική υπάρχει η ταυτότητα του αισθητήρα στον οποίο ανήκει . Η ταυτότητα κάθε μετρικής είναι ένας συνδυασμός της ταυτότητας του αισθητήρα και της μονάδας μέτρησης της μετρικής . Για να είναι επιτεύξιμη η αναγνώριση των δεδομένων που θα αποθηκευτούν υπάρχει ο τύπος δεδομένων που μπορεί να είναι κείμενο, μονοδιάστατο, τοποθεσία κ. α. που έχουν προαναφερθεί . Όπως και στα προηγούμενα όλες οι μεταβλητές μπορούν να ενημερωθούν εκτός από την ταυτότητα τους . Οι μετρήσεις έχουν την ταυτότητα της μετρικής στην οποία ανήκουν , την τιμή value που πρέπει να αποθηκευτεί και την ώρα που πάρθηκε η μέτρηση . Η τιμή value είναι τύπου String ώστε να επιτρέπει την αποθήκευση οποιουδήποτε τύπου δεδομένων χωρίς πρόβλημα και αναγνωρίζεται μέσω της μονάδας μέτρησης και του τύπου δεδομένων .



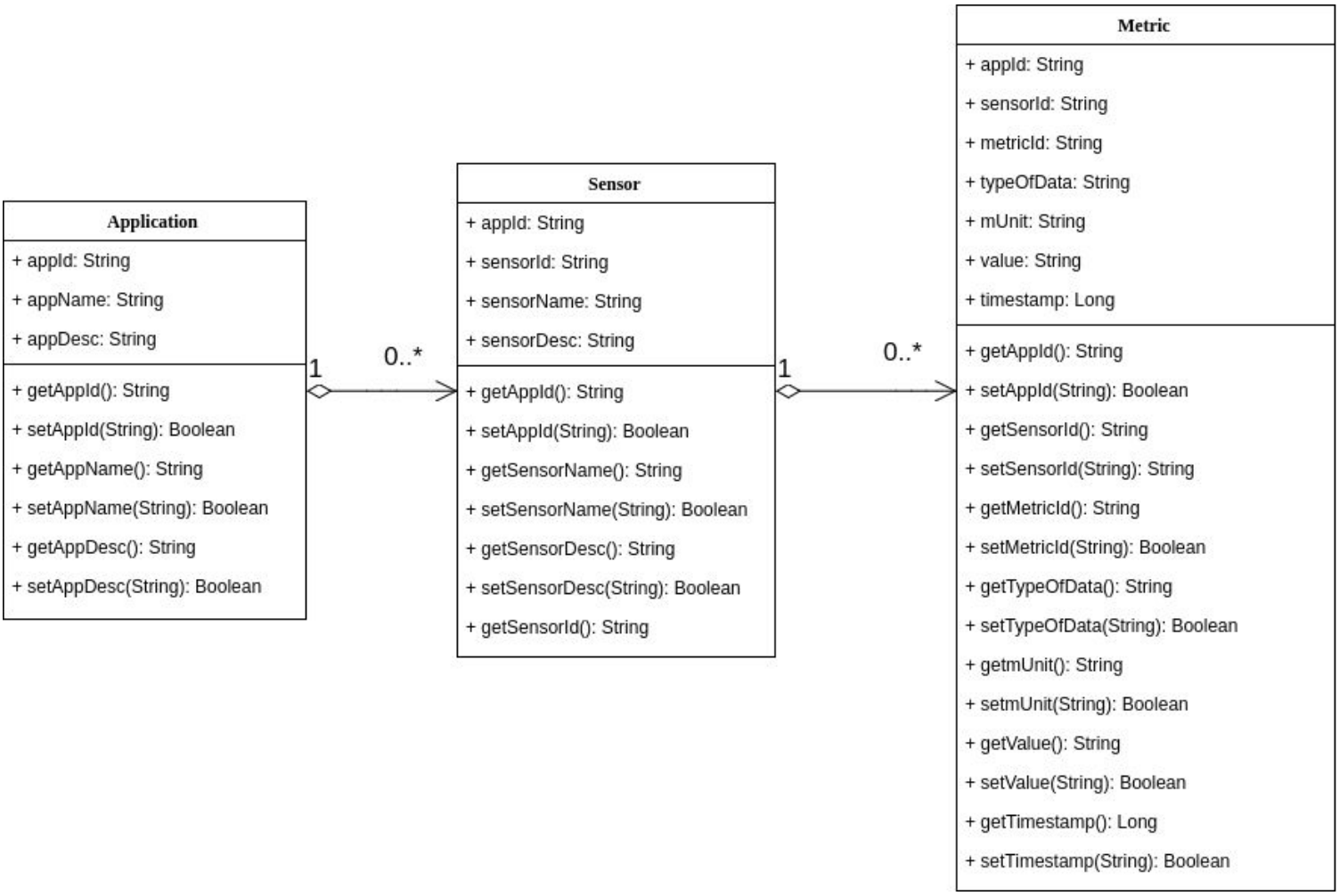
Σχήμα 3.3.1 Σχηματική απεικόνιση μοντέλου δεδομένων .

3.4 Περιγραφή διεπαφής προγραμματισμού εφαρμογών

Η διεπαφή προγραμματισμού εφαρμογών είναι ένα σύνολο από εντολές, λειτουργίες, πρωτόκολλα και αντικείμενα που χρησιμοποιούνται από προγραμματιστές ώστε να δημιουργήσουν λογισμικό ή να αλληλεπιδράσουν με κάποιο εξωτερικό σύστημα . Παρέχει στους προγραμματιστές εντολές με τις οποίες μπορούν να εκτελέσουν κοινές λειτουργίες χωρίς να είναι αναγκαία η συγγραφή κώδικα από την αρχή .

3.4.1 Διεπαφές

Στην διεπαφή προγραμματισμού εφαρμογών υπάρχουν τέσσερις διεπαφές, για την εφαρμογή (Application), τους αισθητήρες (Sensor), τις μετρικές (Metric) και το διαχειριστή βάσης (Db Handler). Οι διεπαφές για εφαρμογή και αισθητήρες υλοποιούνται όπως περιγράφονται στο υποκεφάλαιο ‘μοντέλο δεδομένων’ και στο σχήμα 3.4.1.1 . Για την μετρική και τις μετρήσεις δημιουργήθηκε ένα αντικείμενο όπου η μετρική περιέχει και τις μεταβλητές της μέτρησης. Έτσι όταν θέλουμε να εισάγουμε μετρική θέτουμε το value και timestamp και το στέλνουμε στον διαχειριστή βάσης .



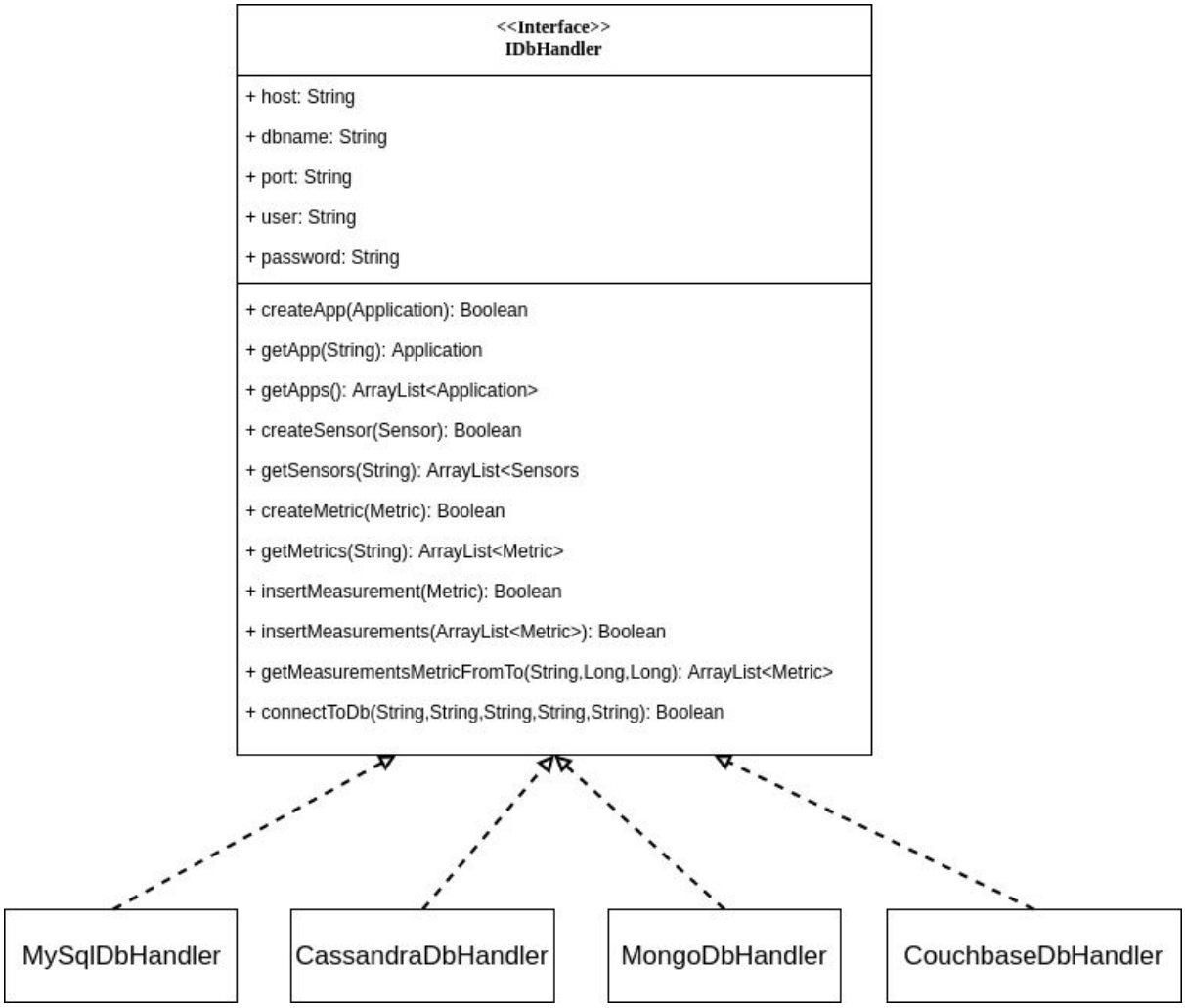
Σχήμα 3.4.1.1 Περιγραφή υλοποίησης εφαρμογής, αισθητήρα και μετρικών .

Η πιο σημαντική διεπαφή είναι η διεπιφάνεια που διαχειρίζεται τις βάσεις δεδομένων IDbHandler . Περιέχει βασικές συναρτήσεις για επικοινωνία και σύνδεση με την βάση όπως και για αποσύνδεση της . Για να συνδεθεί με την βάση αρχικά διαβάζει τις ιδιότητες από το αρχείο iot.properties με την βοήθεια της συνάρτησης readProperties() . Εάν υπάρχουν οι απαραίτητες πληροφορίες γίνεται η σύνδεση με την

βάση . Στο επόμενο σχήμα βλέπουμε ένα παράδειγμα του αρχείου iot.properties .

```
db.host=localhost
db.user=username
db.password=password
db.port=3306
db.name=DBNAME
```

Όπως φαίνεται στο πιο κάτω σχήμα (σχήμα 3.4.1.2) στην διεπαφή υπάρχουν ως συναρτήσεις οι πιο σημαντικές λειτουργίες καθώς και οι πιο συχνές στο περιβάλλον του Internet of Things. Κάθε βάση υλοποιεί την διεπιφάνεια και οποιεσδήποτε συγκεκριμένες λειτουργίες χρειάζονται για την λειτουργικότητα τους.



Σχήμα 3.4.1.2 Περιγραφή διεπαφής διαχείρισης βάσης

Ερωτήματα (queries) που εκτελούνται μέσω της διεπαφής:

- Δημιουργία εφαρμογής (Application)
- Δημιουργία αισθητήρα (Sensor)
- Δημιουργία μετρικής (Metric)
- Εισαγωγή μέτρησης (Measurement)
- Ανάκτηση εφαρμογής (Application)
- Ανάκτηση αισθητήρων (Sensor) με βάση την ταυτότητα της εφαρμογής
- Ανάκτηση μετρικών (Metric) με βάση την ταυτότητα του αισθητήρα
- Ανάκτηση μετρήσεων (Measurement) με βάση την ταυτότητα της μετρικής και συγκεκριμένο χρονικό διάστημα.

3.5 Δημιουργία φόρτου εργασίας

Για τους σκοπούς ελέγχου των βάσεων δημιουργήθηκε πρόγραμμα DataGenerator σε Java το οποίο δέχεται κάποιες εισόδους και στέλνει δεδομένα στις βάσεις . Προσομοιώνει το περιβάλλον του Internet of Things, δημιουργεί εικονικούς αισθητήρες και στέλνει δεδομένα που μπορεί να στείλει οποιοσδήποτε αισθητήρας . Μπορεί να στείλει φόρτο εργασίας μέσω http ή να συνδεθεί απευθείας με τις βάσεις . Ως είσοδο δέχεται τον αριθμό των αισθητήρων και κάθε αισθητήρας είναι ένα αντικείμενο που θα ανοίξει σε νέο νήμα (thread) για να ξεκινήσει την αποστολή δεδομένων . Άλλη παράμετρος είναι ο αριθμός των μετρήσεων που θα στείλει κάθε αισθητήρας, ορίζεται στο μηδέν για να στέλνει συνεχώς μετρήσεις. Επόμενη παράμετρος είναι η περίοδος δηλαδή κάθε πόσο χρονικό διάστημα ο αισθητήρας θα στέλνει δεδομένα. Εάν η παράμετρος της περιόδου είναι 0 θα στέλνει συνεχώς δεδομένα . Έχουμε την επιλογή να ξεκινούν οι αισθητήρες όποτε είναι έτοιμος ο καθένας ή να περιμένουν μέχρι να

δημιουργηθούν όλα τα νήματα (threads) ώστε να ξεκινήσουν την αποστολή όλοι μαζί . Τέλος μπορεί να καθοριστεί η τοποθεσία αποθήκευσης στατιστικών που χρειαζόμαστε καθώς και το όνομα των αρχείων . Για την αποστολή δεδομένων μέσω http χρησιμοποιήθηκε η βιβλιοθήκη Apache Http Client . Επίσης υλοποιήθηκε κλάση thread monitor ώστε να παρακολουθεί την κατάσταση των νημάτων, και να τυπώνει την κατάσταση τους κάθε 5 δευτερόλεπτα (ποια νήματα έχουν ολοκληρωθεί, τρέχουν ή αν περιμένουν στην ουρά) .

Κεφάλαιο 4

Εργαλεία που Χρησιμοποιήθηκαν

4.1 Databases

Όπως έχει ήδη αναφερθεί, για την εξέταση του μοντέλου δεδομένων και framework θα εξεταστούν τέσσερις βάσεις δεδομένων που είναι η MySql, Cassandra, MongoDb και Couchbase . Υπάρχουν δύο τύποι βάσης δεδομένων που θα εξεταστούν, οι relational databases (RDBMS) όπως είναι η MySql και οι NoSql βάσεις όπως είναι η Cassandra, η MongoDb και η Couchbase .

Στην διάθεση μας για εγκατάσταση των βάσεων έχουμε τέσσερις εξυπηρετητές (server) με δύο πυρήνες, 2 gigabytes μνήμη τυχαίας προσπέλασης(RAM) και είκοσι gigabytes μνήμη . Το λειτουργικό σύστημα που είναι εγκατεστημένο στις μηχανές είναι ubuntu 16.04 x86_46 .

Για κάθε βάση θα παρουσιαστεί το σχήμα βάσης δεδομένων (database schema) . Το σχήμα βάσης δεδομένων ορίζει την δομή της βάσης, σε μία σχεσιακή βάση το σχήμα ορίζεται εκ των προτέρων ως πίνακες και σχέσεις μεταξύ οντοτήτων, οποιαδήποτε εισαγωγή οντοτήτων πρέπει να υπακούει στους κανόνες του σχήματος . Στις βάσεις NoSql που υποστηρίζουν μεγάλες ποσότητες δεδομένων διαφορετικών τύπων υπάρχουν βάσεις χωρίς σχήμα ή βάσεις που επιτρέπουν πιο ευέλικτη δομή .

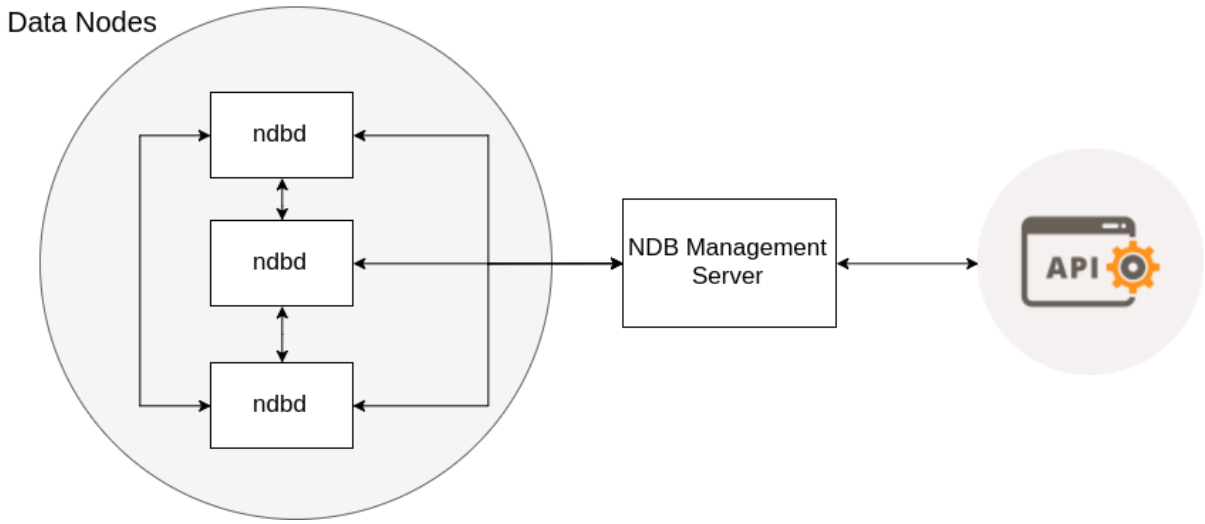
4.1.1 Βάση δεδομένων MySql

Η MySql είναι μια σχεσιακή βάση δεδομένων η οποία αποθηκεύει τα δεδομένα σε διαφορετικούς πίνακες αντί να τα αποθηκεύει σε ένα μεγάλο πίνακα . Η δομή της βάσης είναι οργανωμένη σε αρχεία βελτιστοποιημένα για ταχύτητα. Το λογικό μοντέλο αποτελείται από βάσεις, πίνακες, σειρές και στήλες . Στις σχεσιακές βάσεις δεδομένων υπάρχουν κανόνες με σχέσεις μεταξύ διαφορετικών πεδίων δεδομένων για παράδειγμα ένα προς ένα (one-to-one), ένα προς πολλά (one-to-many) μοναδικό, απαιτούμενο ή προαιρετικό και δείκτες μεταξύ διαφορετικών πινάκων [14] . Η MySql επιβάλλει αυτούς τους κανόνες ώστε με μία καλά σχεδιασμένη βάση να μην υπάρχει ασυνέπεια, διπλότυπα, ορφανά, μη ενημερωμένα ή ελλείποντα δεδομένα . Ο τρόπος πρόσβασης της βάσης γίνεται με SQL (Structured Query Language) που είναι ο πιο διαδεδομένος . Είναι ανοιχτού κώδικα , έτσι είναι δυνατό ο οποιοσδήποτε να χρησιμοποιήσει και να τροποποιήσει το λογισμικό . Η άδεια λογισμικού που χρησιμοποιεί είναι GPL (GNU General Public License) έτσι αν χρειάζεται η πρόσθεση κώδικα σε εμπορική εφαρμογή πρέπει να αγοραστεί ειδική άδεια από την εταιρία.

Υπάρχουν διαφορετικές εκδόσεις MySql όπως είναι MySql Server, MySql Enterprice και MySql NDB Cluster . Για τους σκοπούς εξέτασης της βάσης

εγκαταστάθηκε στους τέσσερις servers που αναφέραμε η έκδοση MySql NDB Cluster . Η συγκεκριμένη έκδοση της MySql είναι υψηλής απόδοσης προσαρμοσμένη για distributed περιβάλλον . Χρησιμοποιεί τον μηχανισμό αποθήκευσης NDB και έτσι το σύστημα μπορεί να λειτουργεί με ελάχιστες απαιτήσεις σε λογισμικό (software) και υλικό (hardware) . Ο NDB cluster ενσωματώνει στην MySql server τον μηχανισμό NDB (Network DataBase) .

Αποτελείται από ένα σύνολο υπολογιστών που ο καθένας μπορεί να είναι MySql server, κόμβος δεδομένων (data nodes) για αποθήκευση δεδομένων ή να είναι διαχειριστής (management server) . Στο σχήμα 4.1.1.1 παρουσιάζεται η αρχιτεκτονική MySql NDB Cluster .



Σχήμα 4.1.1.1 Αρχιτεκτονική MySql NDB Cluster (Βασισμένο στο μοντέλο δεδομένο που προτείνετε πιο πάνω**) .**

Στο επόμενο σχήμα (σχήμα 4.1.1.2) παρουσιάζονται οι τύποι δεδομένων για κάθε χαρακτηριστικό του κάθε πίνακα, τα primary keys και foreign keys .

```
CREATE TABLE Applications (
    appId VARCHAR(36) NOT NULL,
    appName VARCHAR(50) NOT NULL,
    appDescription VARCHAR(300),
    PRIMARY KEY (appId)
);

CREATE TABLE Sensors (
    appId VARCHAR(36) NOT NULL,
    sensorId VARCHAR(36) NOT NULL,
    sensorName VARCHAR(50) NOT NULL,
    sensorDescription VARCHAR(300),
    FOREIGN KEY (appId) REFERENCES Applications(appId),
    PRIMARY KEY (sensorId)
);

CREATE TABLE Metrics (
    appId VARCHAR(36) NOT NULL,
    sensorId VARCHAR(36) NOT NULL,
    metricId VARCHAR(50) NOT NULL,
    typeOfData VARCHAR(30) NOT NULL,
    mUnit VARCHAR(20) NOT NULL,
    PRIMARY KEY (metricId),
    FOREIGN KEY (appId) REFERENCES Applications(appId),
    FOREIGN KEY (sensorId) REFERENCES Sensors(sensorId)
);

CREATE TABLE Measurements (
    metricId VARCHAR(50) NOT NULL,
    value VARCHAR(300) NOT NULL,
    timestamp BIGINT NOT NULL,
    PRIMARY KEY (metricId,timestamp),
    FOREIGN KEY (metricId) REFERENCES Metrics(metricId)
);
```

Σχήμα 4.1.1.2 Σχήμα βάσης δεδομένων .

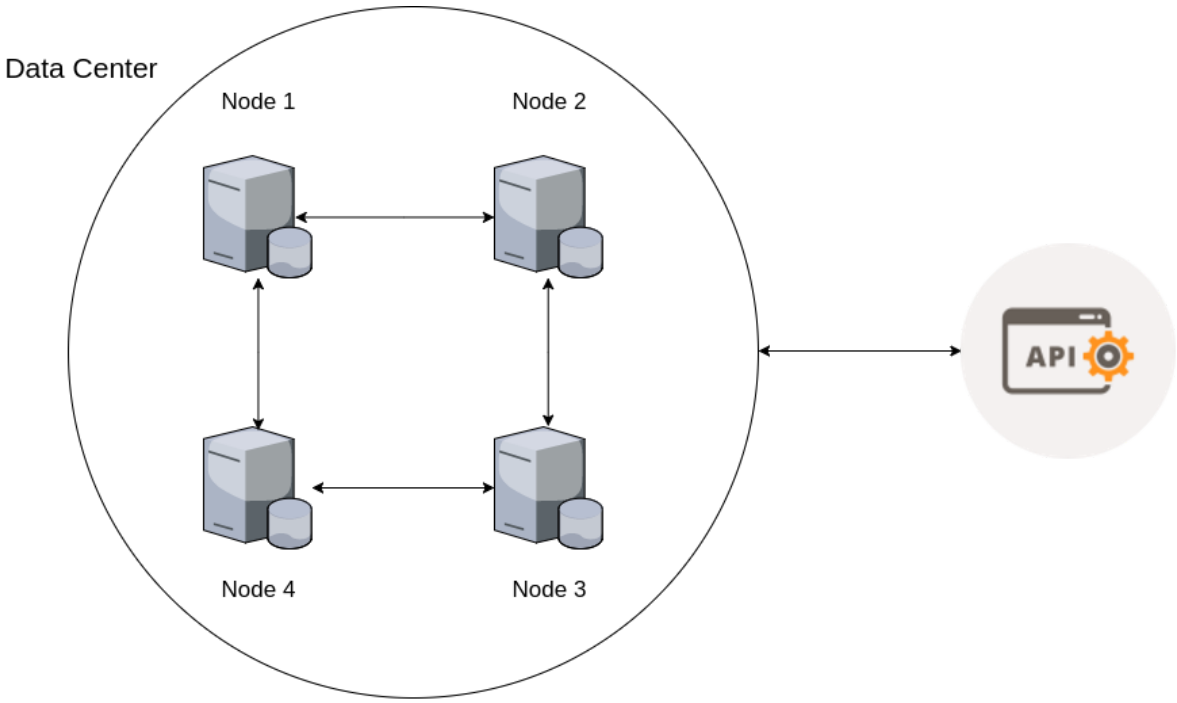
4.1.2 Βάση δεδομένων Cassandra

Η βάση δεδομένων Cassandra είναι ένα κατανεμημένο σύστημα αποθήκευσης δεδομένων χωρίς ενιαίο σημείο αποτυχίας [10] . Έχει ως σκοπό να τρέχει σε υποδομές με εκατοντάδες κόμβους όπου υπάρχουν συνεχώς σφάλματα λειτουργίας από πλευράς υλικού . Σε πολλούς τομείς παρουσιάζει ομοιότητα με σχεσιακές βάσεις δεδομένων και μοιράζεται πολλές στρατηγικές σχεδίασης και υλοποίησης όμως δεν υποστηρίζει πλήρες σχεσιακό μοντέλο . Παρέχει ένα απλό μοντέλο δεδομένων που υποστηρίζει τον δυναμικό έλεγχο και διάταξη της μορφής των δεδομένων [15] . Σχεδιάστηκε ώστε να λειτουργεί σε φθηνό υλικό και να χειρίζεται υψηλή απόδοση εγγραφών (writes) χωρίς να θυσιάζει την απόδοση της ανάγνωσης τους (reads) . Είναι βασισμένη στο Dynamo που είναι key-value σύστημα αποθήκευσης που προτάθηκε από την Amazon . Τα δεδομένα αποθηκεύονται στο mem-table που είναι προσωρινή μνήμη που όταν γεμίσει, τα δεδομένα θα γραφτούν στο SSTable (Sorted String Table) το οποίο είναι ένα αρχείο στον δίσκο . Η διαχείριση δεδομένων μπορεί να γίνει μέσω CQL (Cassandra Query Language) που προσφέρει ένα μοντέλο που μοιάζει με SQL με την έννοια ότι τα δεδομένα μπαίνουν σε πίνακες με γραμμές .

Το μοντέλο δεδομένων της βάσης αποτελείται από πίνακες που είναι διανεμημένοι σε πολλές διαστάσεις με ευρετήριο ένα κλειδί και τα δεδομένα (value) του είναι ένα δομημένο αντικείμενο . Οι στήλες ομαδοποιούνται σε σύνολα που ονομάζονται column-families που χωρίζονται σε απλές και super column families . Οι super column families μπορούν να παρουσιαστούν ως στήλες ως column family μέσα σε column family . Επιπλέον επιτρέπεται οι στήλες να ταξινομούνται με βάση το όνομα ή ώρα (timestamp) . Στο σχήμα 4.1.2.1 βλέπουμε το σχήμα της βάσης δεδομένων όπου φαίνονται οι τύποι δεδομένων και τα πρωτεύοντα κλειδιά . Παρατηρούμε ότι δεν έχουν σχέσεις μεταξύ τους οι πίνακες και οι μετρήσεις είναι ταξινομημένες με βάση το

timestamp . Όλοι οι πίνακες ανήκουν σε ένα keyspace που κατά τη δημιουργία του δηλώνουμε την στρατηγική αναπαραγωγής . Για τις δοκιμές επιλέχθηκε Simple Strategy που επιτρέπει replication factor ακέραιο αριθμό που είναι ο αριθμός των κόμβων που θα περιέχουν αντίγραφο των δεδομένων . Άλλες επιλογές στρατηγικής είναι η Network Topology Strategy και η Tunable Consistency .

Στο σχήμα 4.1.2.1 φαίνεται η αρχιτεκτονική του cluster της βασης δεδομένων. Αποτελείται από δύο seeds που βοηθούν να ανακαλύψουν τα υπόλοιπα nodes χρησιμοποιώντας το gossip protocol για επικοινωνία και ανίχνευση βλαβών. Ένας κόμβος ανταλλάσσει πληροφορίες με άλλους τρεις κόμβους. Οι πληροφορίες για την κατάσταση τους ανταλλάσσονται κάθε δευτερόλεπτο . Οι πληροφορίες έχουν να κάνουν με τον ίδιο τον κόμβο και με τους κόμβους που γνωρίζει .



Σχήμα 4.1.2.1 Αρχιτεκτονική Cassandra Cluster.

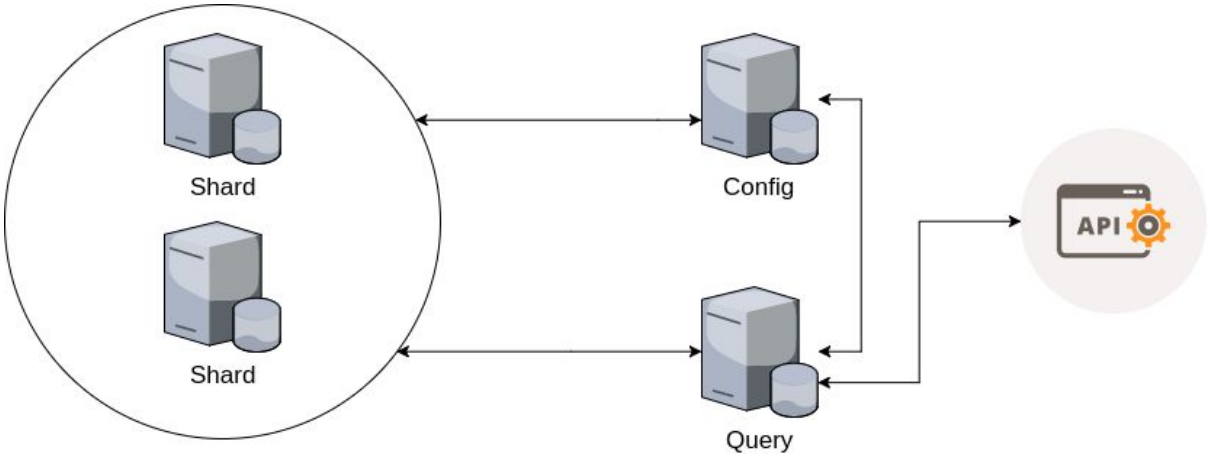
```
CREATE KEYSPACE ADE WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 1};

USE ADE;
CREATE TABLE Applications (appId  varchar, appName  varchar,
    appDescription  varchar, PRIMARY KEY (appId));
CREATE TABLE Sensors (appId  varchar, sensorId  varchar, sensorName  varchar,
    sensorDescription  varchar, PRIMARY KEY (sensorId));
CREATE TABLE Metrics (appId  varchar, sensorId  varchar, metricId  varchar, typeOfData  varchar,
    mUnit  varchar, PRIMARY KEY (metricId));
CREATE TABLE Measurements (metricId  varchar, typeOfData  varchar, mUnit  varchar, value  varchar,
    timestamp bigint, PRIMARY KEY (metricId,timestamp))WITH CLUSTERING ORDER BY (timestamp DESC);
```

Σχήμα 4.1.2.2 Σχήμα βάσης δεδομένων.

4.1.3 Βάση δεδομένων MongoDb

Η βάση δεδομένων MongoDb είναι NoSql σύστημα διαχείρισης δεδομένων,η βάση είναι document-oriented δηλαδή αποθηκεύουν τα δεδομένα σε μορφή αρχείων . Τα αρχεία σε μια document-oriented βάση είναι το αντίστοιχο μιας εγγραφής, γραμμής σε μια σχεσιακή βάση χωρίς όμως να χρειάζονται σχήμα . Τα αρχεία αποθηκεύονται σε μορφή BSON που είναι η εκδοχή του JSON όπου η κωδικοποίηση είναι δυαδική και υποστηρίζει μέχρι δεκαέξι megabytes μέγεθος αρχείου [11] . Υποστηρίζει replicated servers και indexing προσφέροντας drivers για αρκετές γλώσσες προγραμματισμού . Επιπλέον υπάρχει ενσωματωμένη map-reduce συνάρτηση για την διαχείριση μεγάλων ποσοτήτων δεδομένων. Το μοντέλο ακεραιότητας ,αντίστοιχο του ACID στις σχεσιακές βάσεις που χρησιμοποιεί, είναι το BASE που προσφέρει συνέπεια, αντοχή και conditional atomicity [12] αλλά δεν υποστηρίζει απομόνωση (isolation), συναλλαγές (transactions), ακεραιότητα αναφοράς και έλεγχο αναθεώρηση (revision control) . Ο τρόπος επικοινωνίας με την βάση γίνεται με συναρτήσεις για πρόσθεση αρχείων, ενημέρωση και διαγραφή . Τα δεδομένα αποθηκεύονται χρησιμοποιώντας συλλογές (collections) και δεν υπάρχουν περιορισμοί στον τύπο των δεδομένων που αποθηκεύονται ούτε υπάρχει σταθερή δομή, έτσι δίνετε η δυνατότητα στο κάθε πεδίο να έχει διαφορετικό τύπο δεδομένων .

Η βάση δεδομένων MongoDb επιτυγχάνει κλιμάκωση (scaling) χρησιμοποιώντας την τεχνική γνωστή ως “sharding” . Ο όρος sharding είναι η διαδικασία εγγραφής δεδομένων σε διάφορους διακομιστές και να τους διανέμει την το φόρτο για ανάγνωση, εγγραφή και τις απαιτήσεις σε χώρο αποθήκευσης . Το Sharding χωρίζεται σε τρία ξεχωριστά στοιχεία . Έχουμε τον Config Server που σε ένα production system πρέπει να περιέχει τρεις configuration servers . Έτσι εξασφαλίζεται υψηλή αποδοτικότητα . Χρησιμοποιούνται για αποθήκευση μεταδεδομένων (metadata) τα οποία συνδέουν τις αιτήσεις με το shard που το περιέχει και οργανώνει τα δεδομένα . Με αυτόν τον τρόπο οι πληροφορίες μπορούν να ανακτηθούν αξιόπιστα και με συνέπεια. Ένα άλλο στοιχείο που χρειαζόμαστε είναι τα query routers . Πρόκειται για διακομιστές με τους οποίους ενώνεται ο πελάτης . Οι διακομιστές αυτοί είναι υπεύθυνοι για την επικοινωνία με τους configuration servers όπου βρίσκονται τα δεδομένα ώστε να αποκτήσει πρόσβαση και να ανακτήσει τα δεδομένα από τα κατάλληλα shards . Τέλος, οι shard servers είναι υπεύθυνοι για την αποθήκευση δεδομένων. Σε productions systems πρέπει να έχουμε τρεις confuguration servers, δύο query servers και τέσσερις shard servers . Για τα πειράματα όπου έχουμε στην διάθεση μας τέσσερις servers χρησιμοποιήθηκε ένας configuration server, ένας query server και δύο shard servers . Στο σχήμα 4.1.3.1 βλέπουμε την αρχιτεκτονική του cluster . Στο σχήμα 4.1.3.2 βλέπουμε πως δημιουργούνται οι συλλογές, τα ονόματα τους και τις εντολές για διαμοίραση των δεδομένων σε διάφορα shards.



Σχήμα 4.1.3.1 Cluster βάσης δεδομένων .

```
db.createCollection("Applications")
db.createCollection("Sensors")
db.createCollection("Metrics" )
db.createCollection("Measurements")

sh.enableSharding("ADE")
sh.shardCollection("ADE.Applications", { "_id": "hashed" } )
sh.shardCollection("ADE.Sensors", { "_id": "hashed" } )
sh.shardCollection("ADE.Metrics", { "_id": "hashed" } )
sh.shardCollection("ADE.Measurements", { "_id": "hashed" } )
```

Σχήμα 4.1.3.2 Σχήμα βάσης δεδομένων .

4.1.4 Βάση δεδομένων CouchBase

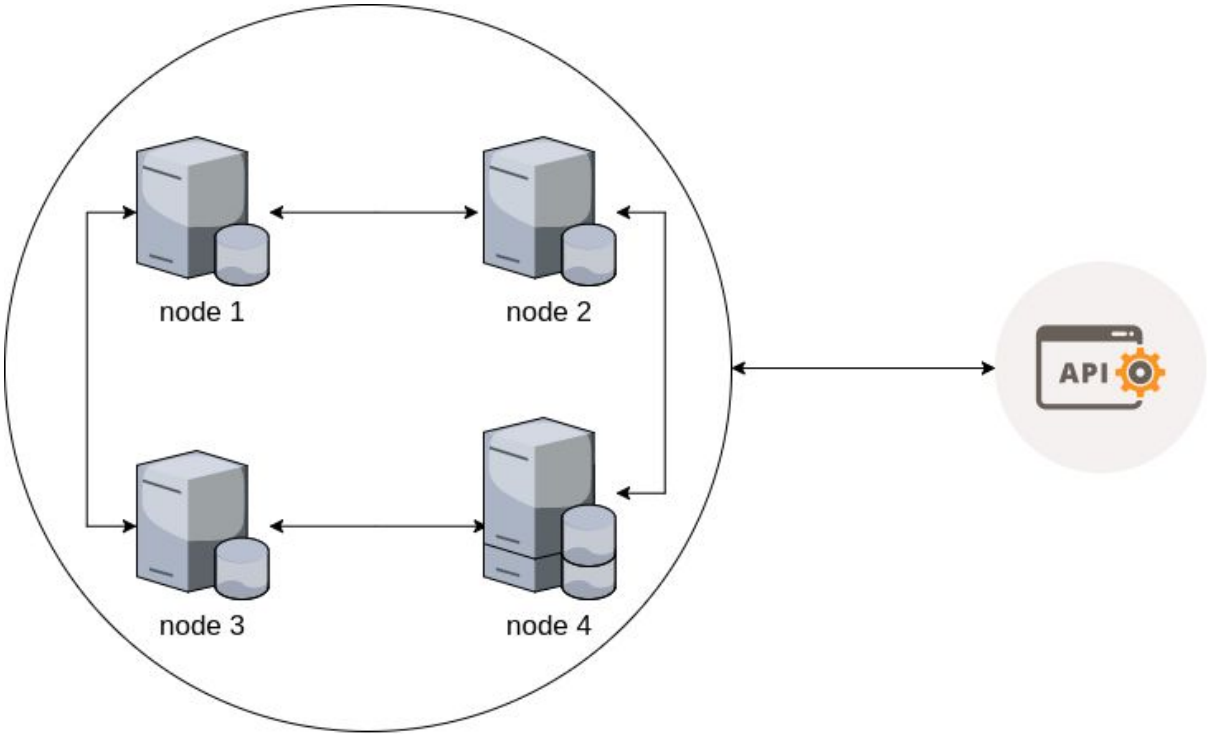
Η Couchbase είναι βάση δεδομένων NoSql ανοιχτού κώδικα που βασίζεται σε έγγραφα όπου για αποθήκευση δεδομένων χρησιμοποιείται key-value store . Το key value store είναι μια απλή προσέγγιση διαχείρισης δεδομένων χωρίς σχήμα βάσης όπου αποθηκεύει ένα μοναδικό κλειδί μαζί με την πληροφορία του (value) . Έτσι διευκολύνονται όλα τα ερωτήματα αφού παρέχετε πρόσβαση δεδομένων μέσω της ταυτότητας τους . Επιπλέον υπάρχουν ευρετήρια (indexers) για πιο γρήγορη απάντηση σε ερωτήματα και μηχανή ερωτημάτων για εκτέλεση εντολών SQL [13] .

Η Couchbase Server κρατά έγγραφα που διαβάζονται συχνά, μεταδεδομένα και ευρετήρια στην RAM ώστε να βελτιώσει τον χρόνο για εγγραφή και ανάγνωση . Τα αρχεία στην Couchbase είναι τύπου JSON όπου δεν υπάρχει προκαθορισμένο σχήμα με πίνακες και στήλες . Κάθε JSON αρχείο έχει το δικό του σχήμα και καθορίζει τα δικά του χαρακτηριστικά . Χωρίς την ύπαρξη κάποιου συγκεκριμένου σχήματος βάσης είναι εφικτό οποιαδήποτε στιγμή να προστεθούν νέα αντικείμενα και ιδιότητες . Κάθε αρχείο αντιπροσωπεύει μια μοναδική εμφάνιση ενός αντικειμένου . Τα αντικείμενα παριστάνουν γραμμές σε ένα σχεσιακό πίνακα που κάθε γραμμή είναι μια εγγραφή .

Η αποθήκευση των δεδομένων γίνεται χρησιμοποιώντας buckets που είναι απομονωμένα εικονικά δοχεία . Είναι μια λογική ομαδοποίηση φυσικών πόρων μέσα σε ένα σύμπλεγμα διακομιστών. Υπάρχουν δύο ειδών buckets, τα memcached που επιτρεπουν την αποθήκευση δεδομένων στην προσωρινή μνήμη μόνο και τα couchbase που επιτρέπουν αποθήκευση στην προσωρινή μνήμη και στον σκληρό δίσκο για επιπρόσθετη αξιοπιστία . Ο τύπος bucket που χρησιμοποιείται στο πείραμα είναι couchbase το οποίο επιτρέπει και μεγαλύτερο μέγεθος αρχείων, αναπαραγωγή και εξισορρόπηση δεδομένων [13] . Η γλώσσα ερωτημάτων (query language) που χρησιμοποιείται ονομάζεται N1QL και έχει αρκετές ομοιότητες με την SQL για

διευκόλυνση των χρηστών . Η N1Q1 είναι ευρέως γνωστή και χρησιμοποιείται από τους περισσότερους οργανισμούς.

Σε κάθε κόμβο στο cluster υπάρχει ο cluster manager που ενορχηστρώνει τις λειτουργίες του . Είναι υπεύθυνος για την τοπολογία του συμπλέγματος και πρόσθεση και αφαίρεση κόμβων, την τοποθεσία δεδομένων, τη συγκέντρωση στατιστικών και αναφορών καθώς και για την επικύρωση συνδέσεων στον cluster . Η αρχιτεκτονική του φαίνεται στο σχήμα 4.1.4.1.



Σχήμα 4.1.4.1 Cluster βάσης δεδομένων .

4.2 Restful services

Το Representational state transfer (REST) είναι ένα αρχιτεκτονικό στυλ για σχεδιασμό κατανεμημένων (distriputed) συστημάτων. Είναι ένα σύνολο από περιορισμούς όπως να υπάρχει η σχέση πελάτη/εξυπηρετητή και μια ομοιόμορφη διεπαφή . Δεν γίνεται οποιαδήποτε αποθήκευση στοιχείων του πελάτη από την πλευρά του εξυπηρετητή κατά την διάρκεια της συνομιλίας . Ο πελάτης έχει τις πληροφορίες για την κατάσταση της συνομιλίας . Οι συνομιλίες γίνονται με μηνύματα μέσω HTTP όπως GET, POST, PUT, DELETE .

4.2.1 OData Rest service

Το Open Data Protocol είναι ένα OASIS standard που καθορίζει τις καλύτερες πρακτικές για κτίσιμο και λειτουργία RESTful APIs . Το OData καθορίζει τις κεφαλίδες αιτήσεων και απαντήσεων, τους κωδικούς κατάστασης, τις μεθόδους HTTP, τις συμβάσεις URL, τη μορφή του φορτίου απάντησης και τις ρυθμίσεις των ερωτημάτων . Καθοδηγεί επίσης τον τρόπο παρακολούθησης αλλαγών, τον ορισμό λειτουργιών και ενεργειών για τις διαδικασίες επαναχρησιμοποίησης και αποστολής ασύγχρονων αιτήσεων . Επιπλέον παρέχει την δυνατότητα επέκτασης για να μπορεί να ικανοποιήσει άλλες ανάγκες που ίσως υπάρξουν .

Για την υλοποίηση του OData Rest service στην java χρησιμοποιήθηκε το Apache Olingo που είναι μια βιβλιοθήκη που υλοποιεί το Open Data Protocol παρέχοντας αρκετές λειτουργίες καθώς και την ίδια τη δομή . Δηλώνοντας όλες τις οντότητες και τα χαρακτηριστικά τους ,ο οποιοσδήποτε, έχει δικαιώματα να ζητήσει αυτές τις πληροφορίες χρησιμοποιώντας $metadata και θα πάρει την απάντηση που φαίνεται στο σχήμα Σχήμα 4.2.1.1 .Στην απάντηση που θα λάβει φαίνεται το όνομα της

οντότητας το κλειδί και οι ιδιότητές της ώστε να γνωρίζει ο χρήστης τον τρόπο εισαγωγής των δεδομένων . Υπάρχει επιλογή για τον τρόπο παρουσίασης της απάντησης η οποία θα είναι είτε XML είτε JSON . Προσθέτοντας την επιλογή $format=application/json στον σύνδεσμο μπορείς να επιλέξεις τον τρόπο παρουσίασης των δεδομένων .Η XML είναι η προκαθορισμένη μορφή .

```
<?xml version='1.0' encoding='UTF-8'?>
 <edmx:Edmx Version="4.0" xmlns:edmx="http://docs.oasis-open.org/odata/ns/edmx">
     <edmx:DataServices>
         <Schema xmlns="http://docs.oasis-open.org/odata/ns/edm" Namespace="OData.Demo">
             <EntityType Name="Application">
                 <Key>
                     <PropertyRef Name="appId"/>
                 </Key>
                 <Property Name="appId" Type="Edm.String"/>
                 <Property Name="appName" Type="Edm.String"/>
                 <Property Name="appDesc" Type="Edm.String"/>
             </EntityType>
             <EntityType Name="Sensor">
                 <Key>
                     <PropertyRef Name="sensorId"/>
                 </Key>
                 <Property Name="appId" Type="Edm.String"/>
                 <Property Name="sensorId" Type="Edm.String"/>
                 <Property Name="sensorName" Type="Edm.String"/>
                 <Property Name="sensorDesc" Type="Edm.String"/>
             </EntityType>
             <EntityType Name="Metric">
                 <Key>
                     <PropertyRef Name="metricId"/>
                 </Key>
                 <Property Name="appId" Type="Edm.String"/>
                 <Property Name="sensorId" Type="Edm.String"/>
                 <Property Name="metricId" Type="Edm.String"/>
                 <Property Name="typeOfData" Type="Edm.String"/>
                 <Property Name="mUnit" Type="Edm.String"/>
                 <Property Name="value" Type="Edm.String"/>
                 <Property Name="timestamp" Type="Edm.Int64"/>
             </EntityType>
```

```
            <EntityType Name="Measurement">
                <Key>
                    <PropertyRef Name="metricId"/>
                </Key>
                <Property Name="appId" Type="Edm.String"/>
                <Property Name="sensorId" Type="Edm.String"/>
                <Property Name="metricId" Type="Edm.String"/>
                <Property Name="typeOfData" Type="Edm.String"/>
                <Property Name="mUnit" Type="Edm.String"/>
                <Property Name="value" Type="Edm.String"/>
                <Property Name="timestamp" Type="Edm.Int64"/>
            </EntityType>
            <EntityContainer Name="Container">
                <EntitySet Name="Applications" EntityType="OData.Demo.Application"/>
                <EntitySet Name="Sensors" EntityType="OData.Demo.Sensor"/>
                <EntitySet Name="Metrics" EntityType="OData.Demo.Metric"/>
                <EntitySet Name="Measurements" EntityType="OData.Demo.Measurement"/>
            </EntityContainer>
        </Schema>
    </edmx:DataServices>
</edmx:Edmx>
```

Σχήμα 4.2.1.1 Απάντηση server σε αίτηση μεταδεδομένων .

Η εισαγωγή δεδομένων γίνεται στέλνοντας ένα JSON αρχείο όπου δίνονται οι απαραίτητες πληροφορίες για την δημιουργία της εγγραφής . Στο σχήμα 4.2.1.2 βλέπουμε την αίτηση και το αποτέλεσμα της . Στον σύνδεσμο δηλώνουμε την οντότητα στην οποία θα στείλουμε την πληροφορία . Στο σχήμα 4.2.1.2 βλέπουμε παράδειγμα δημιουργίας νέου application, το αίτημα αποστολής και της απάντησης από τον server .

http://localhost:8080/IoTDataModelingFramework/DemoService.svc/Application

```
Request:
    {
      "appId":"null",
      "appName":"Weather application",
      "appDesc":"Collects weather conditions."
    }

Answer:
    {
        "@odata.context": "$metadata#Applications",
        "appId": "d66033a32a1e4a0ca9e5c60231b6d4c7",
        "appName": "Weather application",
        "appDesc": "Collects weather conditions."
    }
```

Σχήμα 4.2.1.2 Εισαγωγή εφαρμογής στην βάση .

Το Odata προσφέρει ένα σύστημα ανάλυσης ερωτημάτων . Την επιλογή $filter=<BooleanExpression> . Το πρόγραμμα αναλύει τον σύνδεσμο, φέρνει όλα τα δεδομένα της οντότητας, εφαρμόζει πάνω τους το ερώτημα (query) που είναι το boolean expression που παίρνει ως είσοδο από το $filter, και στέλνει πίσω την απάντηση . Για το περιβάλλον του Internet of Things δεν είναι αποδοτική λύση αφού μπορεί να υπάρχουν οντότητες με εκατομμύρια εγγραφές και κοστίζει να τις φέρνουμε στην μνήμη συνεχώς . Αφού για να εφαρμόσει το ερώτημα πρέπει να φέρει από την βάση όλα τα δεδομένα . Έτσι χρησιμοποιώντας τις πρακτικές Odata υλοποιήθηκε συνάρτηση που να αναλύει τον σύνδεσμο, να δημιουργεί το ερώτημα που ζητήθηκε και να το στέλνει στην βάση ώστε να φορτώσει μόνο τα ζητούμενα και τα δεδομένα και να στείλει την απάντηση .

Στο σχήμα 4.2.1.3 βλέπουμε τις δυνατότητες που προσφέρει η διεπαφή προγραμματισμού εφαρμογών . Στην στήλη path φαίνεται η το κείμενο που πρέπει να προστεθεί στον σύνδεσμο . Στην δεύτερη στήλη βλέπουμε την μέθοδο http που θα σταλεί ο σύνδεσμος και στην τρίτη στήλη την απάντηση από τον διακομιστή .

Path	Method	Response
/$metadata	GET	Returns metadata for all entities (application, sensor,metric, measurement)
/Applications	GET	Returns details about every application in the database
/Sensors?$filter=appId eq 'appId value'	GET	Returns sensors that belongs to the specific application
/Metrics?$filter=sensorId eq 'sensorId value'	GET	Returns metrics tha belongs to the specific sensor
/Measurements?$filter=metricId eq 'metricId value' and timestamp gt 'timestamp 1' and timestamp lt 'timestamp 2'	GET	Returns measurements with timestamp value between the two timestamps for the specific metric
/Applications?data=Json	POST	Creates new application
/Sensors?data=Json	POST	Creates new sensor
/Metrics?data=Json	POST	Creates new metric
/Measurements?data=Json	POST	Insert a new measurement

Σχήμα 4.2.1.3 Περιγραφή API .

4.3 Apache Tomcat

Ο Tomcat είναι ένας διακομιστής εφαρμογών που εκτελεί java servlets (δηλαδή ειδικά προγράμματα που τρέχουν σε διακομιστές) . Στη συνέχεια θα περιγραφεί ο τρόπος με τον οποίο χρησιμοποιήθηκε ο Tomcat . Γίνεται εξαγωγή της διεπαφής προγραμματισμού εφαρμογών και τοποθετείται στον Tomcat ώστε να μπορεί να απαντά στα αιτήματα . Για να μπορεί ο tomcat server να ανοίξει όσο το δυνατό περισσότερα threads , χρειάστηκαν αλλαγές στο αρχείο servel.xml . Στο αρχείο βρίσκονται οι βασικές ρυθμίσεις του tomcat και είναι υπεύθυνο για τις αρχικές ρυθμίσεις εκκίνησης και λειτουργίας του tomcat . Υπάρχουν πέντε βασικές κατηγορίες top level elements: connectors, containers, nested components, και global settings . Οι αλλαγές έγιναν στα connectors όπως φαίνονται στο σχήμα 4.3.1 . Ο μέγιστος αριθμός από threads αυξήθηκε σε 32000 που είναι το μέγιστο του λειτουργικού συστήματος . Μετά από αυτή την αλλαγή χρειάστηκε να αυξηθεί ο αριθμός των ανοικτών αρχείων στο λειτουργικό σύστημα χρησιμοποιώντας ulimit -n 65536 . Για να μην σταματά τις συνδέσεις ο connector χρειάστηκε να θέσουμε το πρωτόκολλο όπως φαίνεται στο σχήμα .

```
<Connector port="80"
       connectionTimeout="200000"
       redirectPort="8443"
       protocol="org.apache.coyote.http11.Http11NioProtocol"
       maxThreads="32000"
   />
```

Σχήμα 4.3.1 Tomcat settings .

Κεφάλαιο 5

Πειράματα και αποτελέσματα

5.1 Αριθμός αιτημάτων που ικανοποιούνται το δευτερόλεπτο

5.2 Χρόνος που χρειάζεται για εγγραφή σε σχέση με τον αριθμό των μετρικών που στέλνουν δεδομένα

5.3 Χρόνος που χρειάζεται από την στιγμή που στέλνουμε τα δεδομένα μέχρι να είναι έτοιμα για παρουσίαση

5.4 Χρόνος που χρειάζεται για να διαβαστεί μια εγγραφή όταν οι αιτήσεις είναι 50% για ανάγνωση και 50% εγγραφή

5.5 Χρόνος που χρειάζεται για να αποθηκευτεί μια εγγραφή όταν οι αιτήσεις είναι 50% για ανάγνωση και 50% εγγραφή

5.6 Αριθμός ανοιχτών συνδέσεων

5.7 Πείραμα με τυχαία περίοδο

5.8 Ανάγνωση ενός εκατομμυρίου εγγραφών όσο αυξάνονται οι αποθηκευμένες εγγραφές στην βάση

5.9 Ανάκτηση εγγραφών

5.10 Έλεγχος χρόνου δημιουργίας σύνδεσης με την βάση.

Για τα πιο κάτω πειράματα χρησιμοποιήθηκαν τέσσερις εικονικές μηχανές (virtual machines) με ubuntu 16.04 και χαρακτηριστικά δύο επεξεργαστές, δύο gigabyte μνήμη τυχαίας προσπέλασης και είκοσι gigabyte μνήμη σκληρού δίσκου για την εγκατάσταση των βάσεων δεδομένων. Χρησιμοποιήθηκε μια εικονική μηχανή ubuntu

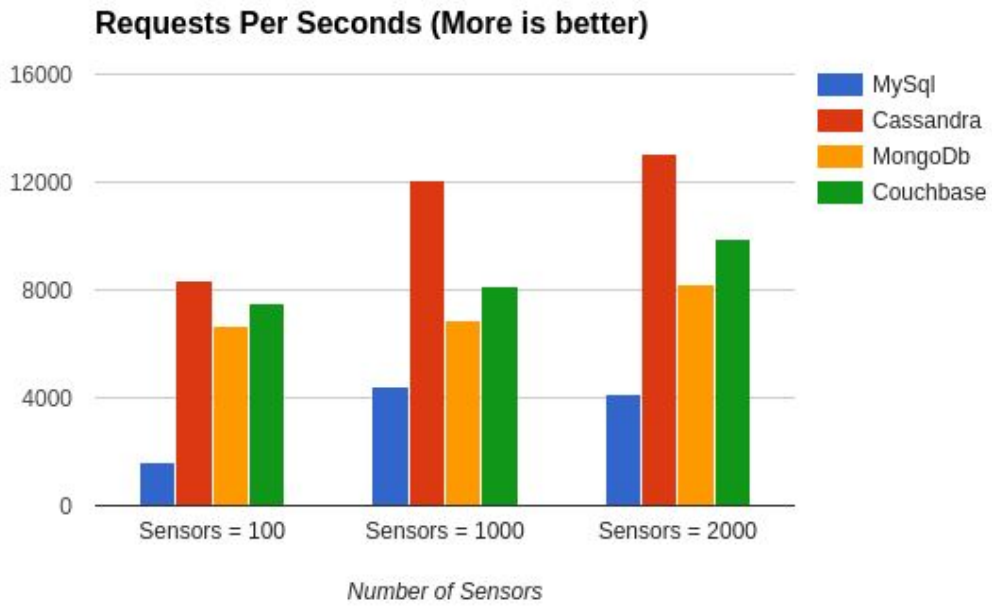
16.04 με οκτώ επεξεργαστές τέσσερα gigabyte μνήμη τυχαίας προσπέλασης και σαράντα gigabyte μνήμη σκληρού δίσκου όπου εγκαταστάθηκε ο tomcat server μαζί με το restful service και την διεπαφή προγραμματισμού εφαρμογών. Στην ίδια μηχανή παράγονται τα δεδομένα μέσω του προγράμματος που δημιουργήθηκε και όπου με http αιτήματα στέλνονται στην διεπαφή προγραμματισμού εφαρμογών που είναι εγκατεστημένη στον tomcat server και τα στέλνει στις βάσεις δεδομένων.

Εκδόσεις εργαλείων που χρησιμοποιήθηκαν:

- MySql Cluster 5.5.5, java connector 6.0.5
- Apache Cassandra 3.10, java connector 3.1.3
- MongoDb 3.2, java connector 3.4.2
- Couchbase community 4.5.1, java connector 2.4.4
- Restful Odata 4.0.0
- Tomcat 8
- Maven version 4.0.0

5.1 Αριθμός αιτημάτων που ικανοποιούνται το δευτερόλεπτο

Σε αυτό το πείραμα υπολογίζεται ο αριθμός τον ολοκληρωμένων αιτήσεων για εγγραφή, με ένα σταθερό αριθμό αισθητήρων που στέλνουν συνεχώς μετρήσεις . Όταν ολοκληρωθεί μια μέτρηση στέλνεται η επόμενη και κάθε αισθητήρας στέλνει τον ίδιο αριθμό από μετρήσεις .

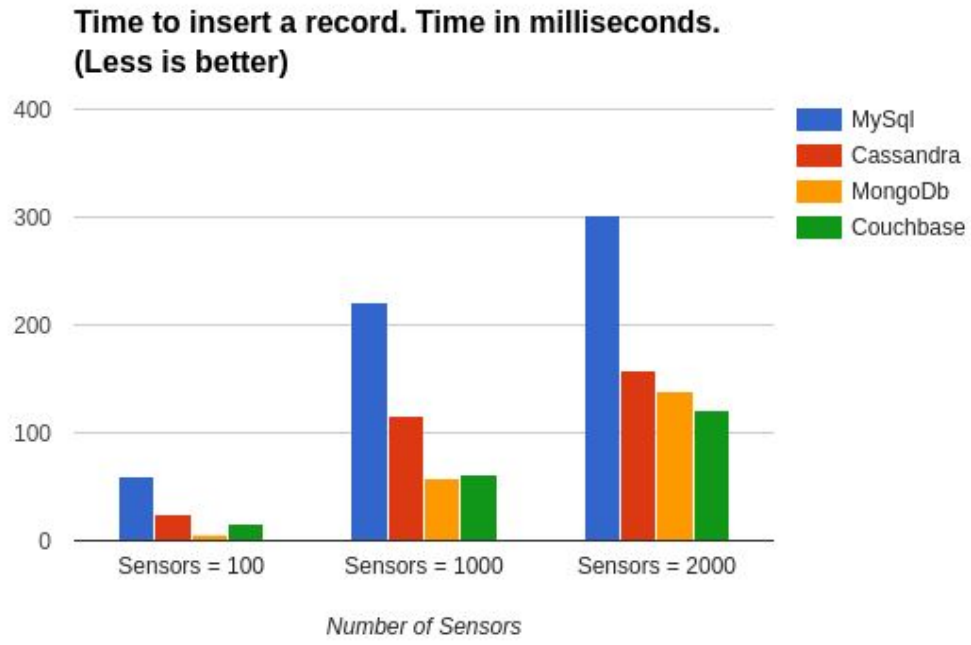


Σχήμα 5.1.1 Αιτήματα το δευτερόλεπτο

Αρχικά για το σχήμα Σχήμα 5.1.1 Πείραμα 1 παρατηρούμε ότι η MySql έχει αρκετά χαμηλό αριθμό ολοκληρωμένων αιτημάτων κάθε δευτερόλεπτο σε σχέση με τις NoSql βάσεις . Η MySql στις 1000 μετρικές έχει τον ψηλότερο αριθμό ολοκλήρωσης αιτημάτων χωρίς να συμπεριλαμβάνονται οι NoSql, όπου για μεγαλύτερο αριθμών μετρικών φαίνεται να μειώνεται ο αριθμός ολοκλήρωσης αιτημάτων . Για μικρό αριθμό μετρικών όλες οι NoSql βάσεις φαίνεται να είναι αρκετά κοντά αλλά όσο αυξάνεται ο αριθμός τους δίνει καλύτερα αποτελέσματα η Cassandra db και μετά Couchbase . Η Cassandra φαίνεται να διαχειρίζεται αρκετά καλά τα αιτήματα χωρίς να παρουσιάζει οποιαδήποτε δυσκολία όπως και η MongoDb και Couchbase αλλά όχι τόσο αποδοτικά .

5.2 Χρόνος που χρειάζεται για εγγραφή σε σχέση με τον αριθμό των αισθητήρων που στέλνουν δεδομένα

Σε αυτό το πείραμα αυξάνεται ο αριθμός των αισθητήρων που στέλνουν δεδομένα και συλλέγεται ο χρόνος που χρειάζεται να αποθηκευτεί μια εγγραφή . Οι αισθητήρες στέλνουν ένα σταθερό αριθμό από μετρήσεις, ενώ ο αριθμός των αισθητήρων αυξάνεται .

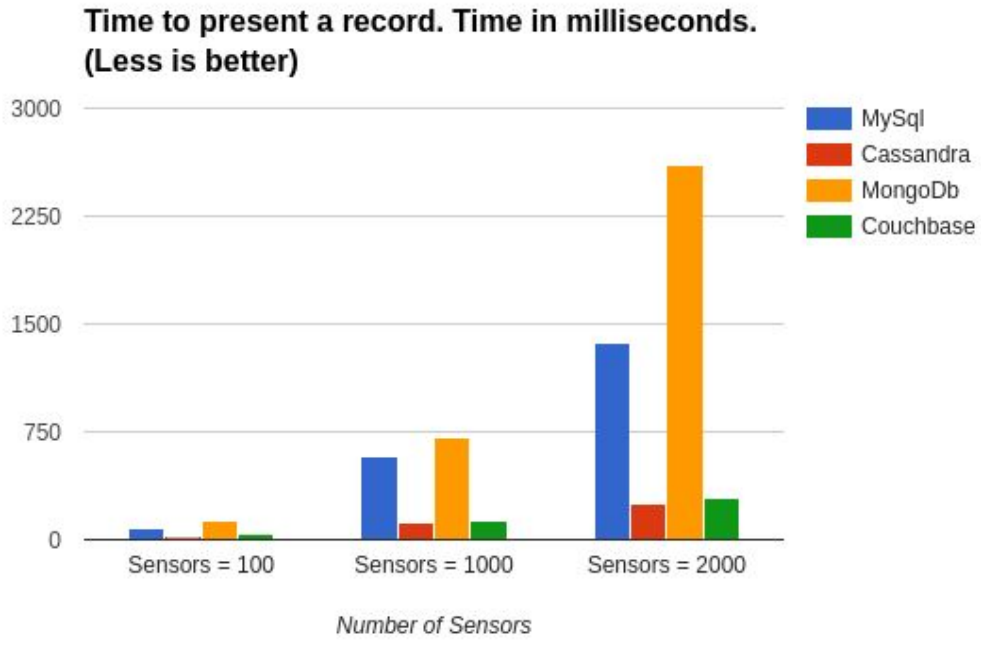


Σχήμα 5.2.1 Χρόνος εγγραφής

Στο σχήμα 5.2.1 Πείραμα 2 παρατηρείται ότι οι document oriented βάσεις αποδίδουν καλύτερα στην απλή εισαγωγή δεδομένων αλλά βρίσκεται αρκετά κοντά και η Cassandra, με το χειρότερο αποτέλεσμα να το έχει η MySql με διπλάσιο χρόνο εισαγωγής από τις NoSql βάσεις δεδομένων.

5.3 Χρόνος που χρειάζεται από την στιγμή που στέλνουμε τα δεδομένα μέχρι να είναι έτοιμα για παρουσίαση

Αποθηκεύεται ο χρόνος που χρειάζεται ο αισθητήρας για να στείλει την αίτηση και ο χρόνος που η μέτρηση είναι έτοιμη για παρουσίαση και υπολογίζεται η διαφορά τους . Ο χρόνος παρουσίασης της μέτρησης είναι σε δέκατα του δευτερολέπτου .

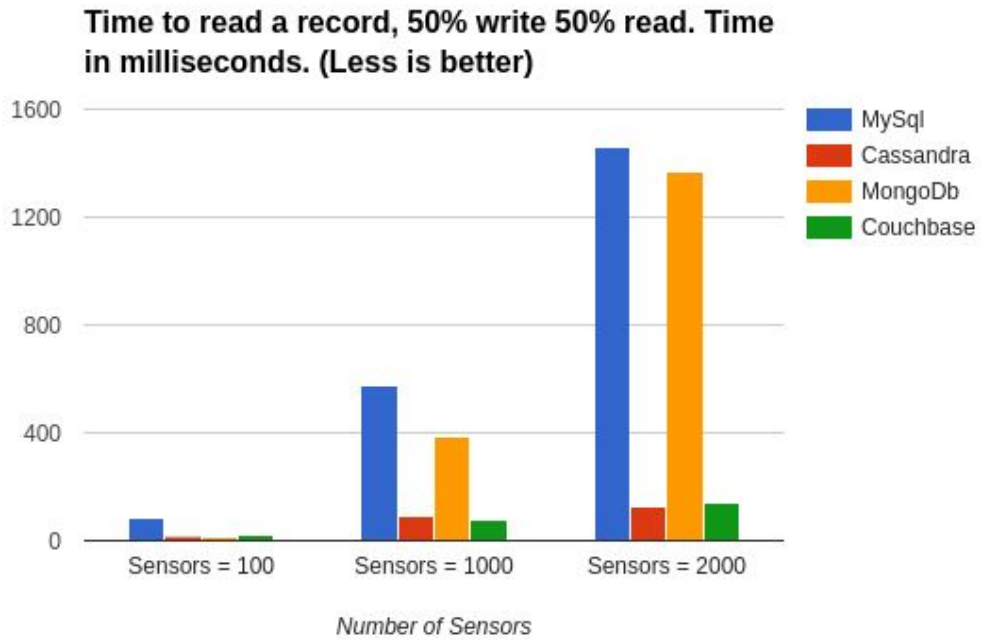


Σχήμα 5.3.1 Χρόνος μέχρι την δυνατότητα παρουσίασης δεδομένων

Στον συνδυασμό γράψιμο εγγραφής και ανάγνωσης της αποδίδει καλύτερα η CouchBase για μεγαλύτερο αριθμό μετρικών και βρίσκεται αρκετά κοντά η CassandraDb . Η MongoDb έχει την χειρότερη απόδοση από τις NoSql βάσεις με αποτελέσματα κοντά στην MySql .

5.4 Χρόνος που χρειάζεται για να διαβαστεί μια εγγραφή όταν οι αιτήσεις είναι 50% για ανάγνωση και 50% εγγραφή

Για κάθε μέτρηση που στέλνεται από τον αισθητήρα , στέλνεται και μια αίτηση για ανάγνωση της εγγραφής . Ο χρόνος είναι σε δέκατα του δευτερολέπτου .

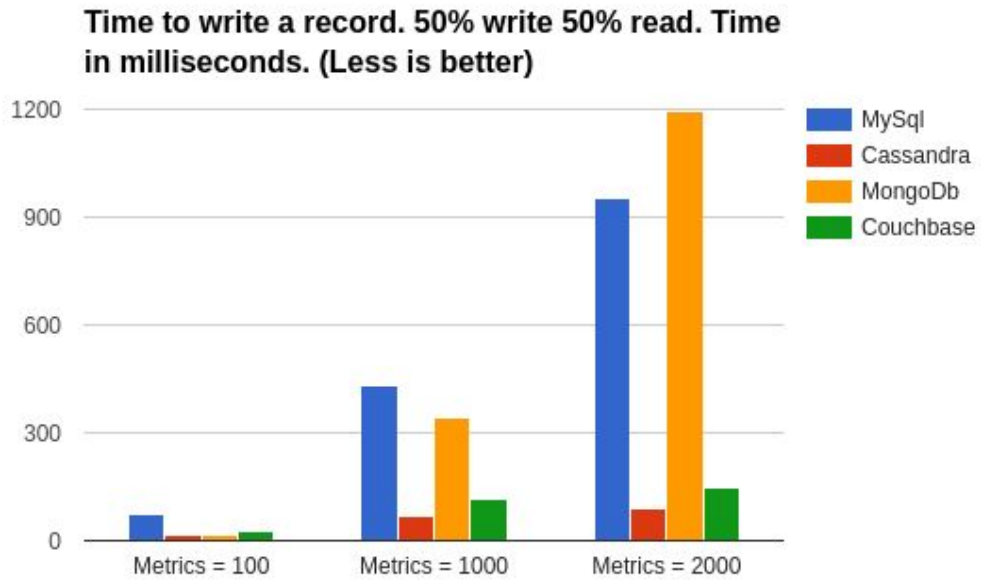


Σχήμα 5.4.1 Χρόνος Ανάγνωσης

Παρατηρείται στο σχήμα Σχήμα 5.4.1 Πείραμα 4 ότι καθώς αυξάνονται οι μετρικές η Cassandra και Couchbase διαχειρίζονται αρκετά αποδοτικά την ανάγνωση των εγγραφών, αντίθετα Mongodb παρουσιάζει σχεδόν ίδια αποτελέσματα με την MySql .

5.5 Χρόνος που χρειάζεται για να αποθηκευτεί μια εγγραφή όταν οι αιτήσεις είναι 50% για ανάγνωση και 50% εγγραφή .

Για κάθε μέτρηση που στέλνεται από τον αισθητήρα στέλνεται και μια αίτηση για ανάγνωση της εγγραφής . Ο χρόνος είναι σε δέκατα του δευτερολέπτου .

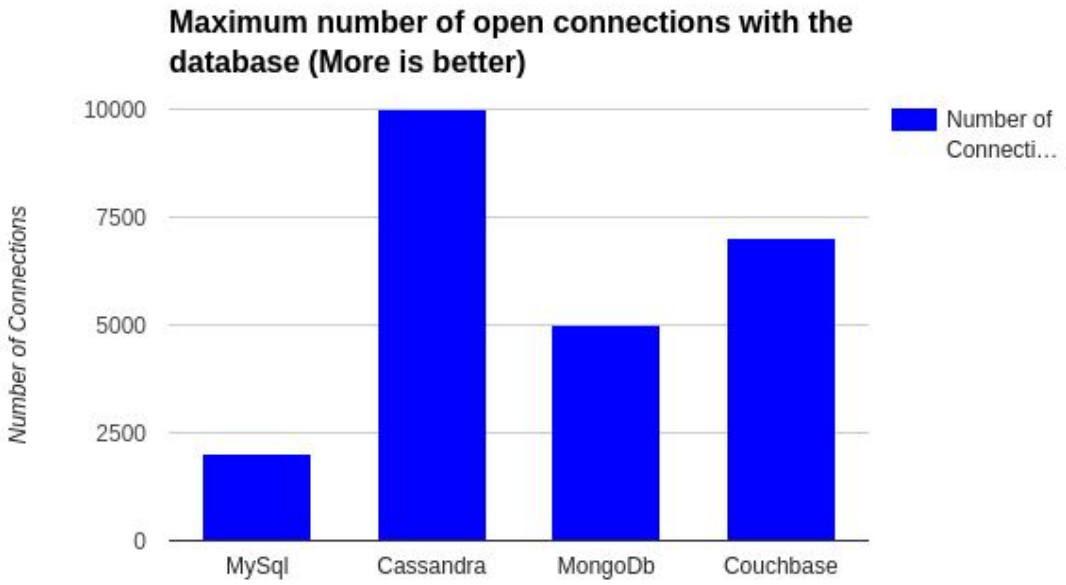


Σχήμα 5.5.1 Χρόνος εγγραφής

Στο πείραμα Σχήμα 5.5.1 Πείραμα 5 παρουσιάζονται περίπου τα ίδια αποτελέσματα με το πείραμα 4 όπου είναι το ίδιο σενάριο αλλά βλέπουμε τον χρόνο εγγραφής των δεδομένων . Σε αυτή την περίπτωση η MongoDb φαίνεται να έχει μεγαλύτερη καθυστέρηση εγγραφής και από την MySql .

5.6 Αριθμός ανοιχτών συνδέσεων.

Αριθμός των ανοιχτών συνδέσεων που μπορούν να διαχειριστούν οι βάσεις . Για αυτό το πείραμα αυξανόταν συνεχώς ο αριθμός των αισθητήρων που στέλνουν μετρικές στην βάση μέχρι η βάση να μην μπορεί να τις διαχειριστεί .

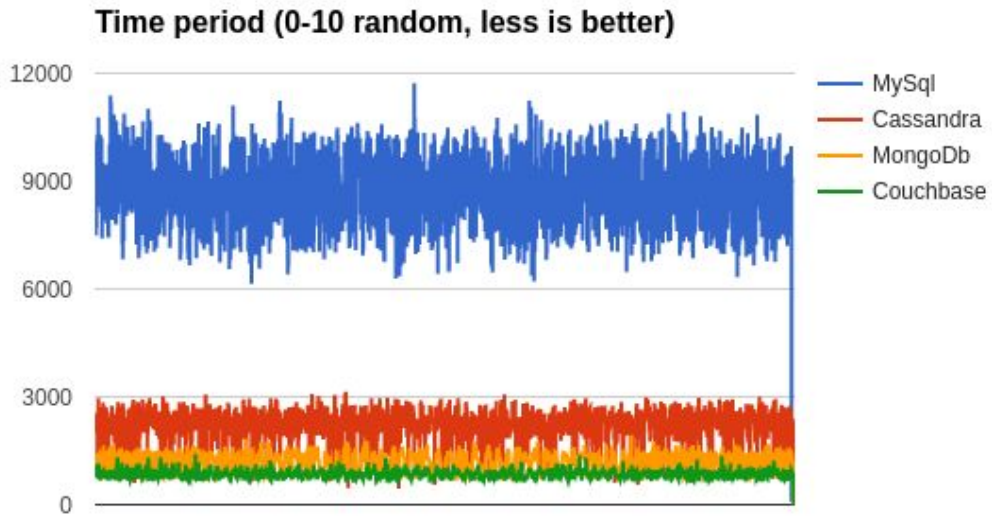


Σχήμα 5.6.1 Αριθμός συνδέσεων

Στο πείραμα 6 φαίνεται ο αριθμός ανοικτών συνδέσεων που μπορούν να διαχειριστούν οι βάσεις χωρίς κανένα πρόβλημα .

5.7 Πείραμα με τυχαία περίοδο .

Γίνεται αποστολή δεδομένων από αισθητήρες με περίοδο τυχαία από μηδέν μέχρι δέκα δευτερόλεπτα με σταθερό αριθμό αισθητήρων και σταθερό αριθμό μετρικών . Ο χρόνος που παρουσιάζεται είναι ο χρόνος εισαγωγής των μετρήσεων για κάθε αισθητήρα. (ms)

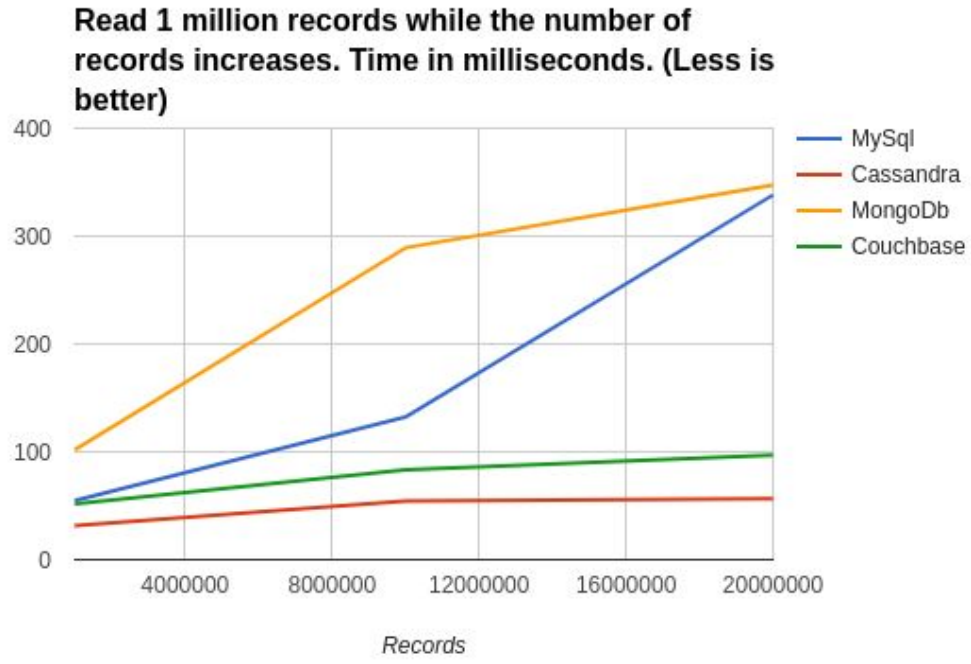


Σχήμα 5.7.1 Αριθμός εγγραφής με τυχαία περίοδο

Στο πείραμα 7 παρατηρούμε τον μέσο χρόνο εισαγωγής δεδομένων δεδομένων από κάθε μετρική με Couchbase και MongoDb να έχουν τα καλύτερα αποτελέσματα, λίγο πιο αργή η CassandraDb και με αρκετή καθυστέρηση η MySql .

5.8 Ανάγνωση ενός εκατομμυρίου εγγραφών όσο αυξάνονται οι αποθηκευμένες εγγραφές στην βάση.

Σε κάθε προσπάθεια στέλνουμε αίτημα ανάγνωσης ενός εκατομμυρίου εγγραφών από την βάση όταν οι αποθηκευμένες εγγραφές είναι ένα εκατομμύριο, δέκα εκατομμύρια και είκοσι εκατομμύρια .

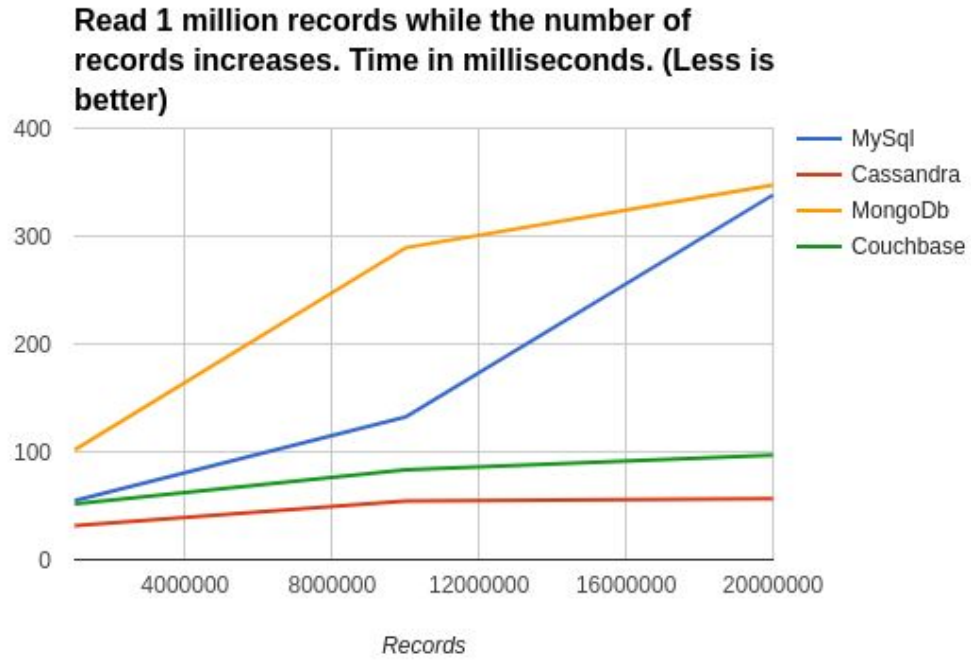


Σχήμα 5.8.1 Ανάγνωση εγγραφών με διαφορετικό αριθμό αποθηκευμένων δεδομένων.

Όσο αυξάνεται ο αριθμός των εγγραφών η CassandraDb μπορεί να διαχειριστεί καλύτερα την ανάγνωση των δεδομένων όπως και η Couchbase . Ενδιαφέρον είναι το ότι σε αυτό το πείραμα η MySql κατάφερε να αποδώσει καλύτερα από την MongoDb .

5.9 Ανάκτηση εγγραφών

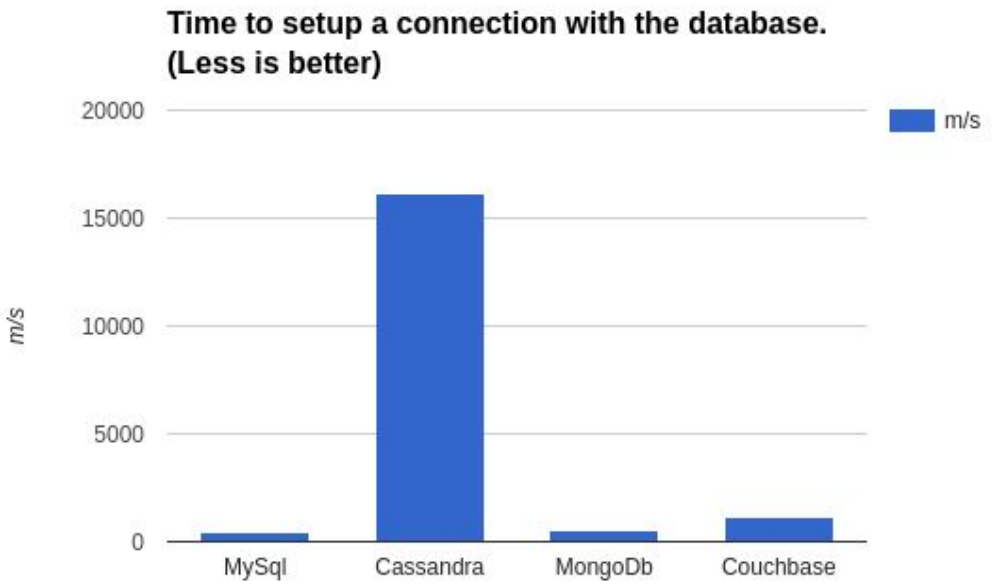
Σε αυτό το πείραμα ελέγχεται ο χρόνος ανάκτησης, 10000 εγγραφών, 100000 και 1000000.



Σχήμα 5.9.1 Ανάγνωση διαφορετικού αριθμού εγγραφών

Όπως και στο πείραμα 8 βλέπουμε την ίδια απόδοση από τις βάσεις για ανάγνωση με καλύτερη απόδοση η CassandraDb και χειρότερη η MySql .

5.10 Έλεγχος χρόνου δημιουργίας σύνδεσης με την βάση.



Σχήμα 5.10.1 Χρόνος δημιουργίας σύνδεσης με βάση.

Στο πιο πάνω σχήμα φαίνεται ο χρόνος δημιουργίας σύνδεσης με την βάση . Αξίζει να σημειωθεί ότι οι NoSql βάσεις προσφέρουν pool connections για αυτό γίνεται μια φορά η σύνδεση ενώ η MySql για κάθε νέα σύνδεση χρειάζεται τον πιο πάνω χρόνο για αυτό δίνει και τα καλύτερα αποτελέσματα .

Κεφάλαιο 6

Συμπεράσματα

6.1 Προβλήματα που αντιμετωπίστηκαν

Στην προσπάθεια δημιουργίας cluster για την MySql παρουσιάστηκε το πρόβλημα ότι οι μηχανές ndbd δεν μπορούσαν να συνδεθούν στον managment server . Το πρόβλημα προέκυψε γιατί στο αρχείο config.ini δηλώθηκαν τρία ndbd ως data nodes και ένας managment server . Μετά από διερεύνηση ανακαλύφτηκε ότι ο αριθμός των ndbd nodes πρέπει να είναι σε δύναμη του δύο (π.χ. 2,4,8…) , έτσι ο managment server προστέθηκε ως data node αλλά παρέμεινε απενεργοποιημένος . Επιπλέον ο java mysql connector δεν υποστηρίζει pool connections έτσι δεν μπορούσε να ανοίξει αρκετές συνδέσεις με τον server . Για την επίλυση του πιο πάνω θέματος δημιουργήθηκε πρόγραμμα για να διαχειρίζεται τις συνδέσεις χρησιμοποιώντας Generic Object Pool . Επιπλέον για βελτίωση των αποτελεσμάτων δημιουργήθηκε ευρετήριο στον πίνακα Measurements και παρουσιάστηκε μικρή βελτίωση στα αποτελέσματα .

Η MongoDb και CouchBase υποστήριζαν pool connections όμως πάλι δεν μπορούσαν να ανοίξουν αρκετές συνδέσεις με τον server . Αυτό το πρόβλημα οφειλόταν στο ότι οι βάσεις είναι document-oriented με αποτέλεσμα ο αριθμός ανοικτών συνδέσεων να επηρεάζεται από τον αριθμό ανοικτών αρχείων που επιτρέπουν τα linux

ubuntu . Όπως και στον tomcat server το πρόβλημα διορθώθηκε αλλάζοντας τον αριθμό των ανοιχτών αρχείων με την εντολή ulimit -n 65536 σε όλους τους servers .

Ο προτεινόμενος αριθμός servers για την δημιουργία cluster με την βάση MongoDb είναι εννέα και , λόγω του ότι ο διαθέσιμος αριθμός server για τα πειράματα ήταν τέσσερις servers, παρατηρήθηκαν κάποια προβλήματα . Τα shards , που όπως αναφέραμε αποθηκεύουν τα δεδομένα , είχαν συνεχώς 100% λειτουργία επεξεργαστή . Επίσης από τους τρεις configuration servers που είναι το προτεινόμενο, για το πείραμα χρησιμοποιήθηκε ένας . Ακόμα χρησιμοποιήθηκε ένας query server, ενώ ο προτεινόμενος αριθμός είναι δύο . Για βελτίωση αποτελεσμάτων δημιουργήθηκε index στην συλλογή Measurements όπου υπάρχει φθίνουσα ταξινόμηση σε σχέση με το timestamp . Βελτιώθηκαν τα αποτελέσματα όχι όμως αρκετά ώστε να μας δώσει καλύτερα αποτελέσματα από την MySql .

Η Couchbase δεν υποστηρίζει ubuntu 16.04 , που είναι το λειτουργικό σύστημα που υπάρχει στις μηχανές που χρησιμοποιήθηκαν, υποστηρίζει μέχρι ubuntu 14.04 . Επίσης παρουσιάστηκαν κάποια προβλήματα με την εγκατάσταση και τον τρόπο εκκίνησης της βάσης αφού με την επίσημη εγκατάσταση δεν ήταν εφικτή η δημιουργία του service στο λειτουργικό και ούτε άνοιγαν οι κατάλληλες θύρες και υπήρξε η ανάγκη επανεγκατάσταση της αρκετές φορές αφού δεν μπορούσε να λειτουργήσει .

Αυτά είναι τα προβλήματα που αντιμετωπίστηκαν κατά τη διεξαγωγή των πειραμάτων . Κάποια από τα προβλήματα επηρέασαν τα εξαγόμενα αποτελέσματα όμως στην πλειοψηφία τους τα προβλήματα περιορίστηκαν στη δυσχέρεια διεξαγωγής των πειραμάτων .

6.2 Συμπεράσματα πειραμάτων

Με βάση τις παρατηρήσεις που κάναμε σε σχέση με τα αποτελέσματα των πειραμάτων συμπεραίνουμε ότι η βάση MySql έδωσε αποτελέσματα τα οποία κυμαίνονται στα επίπεδα που θα περιμέναμε . Σε κάθε πείραμα είναι πιο αργή από τις NoSql βάσεις εκτός από την MongoDb που στις περισσότερες περιπτώσεις βρίσκονται στα ίδια επίπεδα. Η MongoDb έχει χειρότερο χρόνο στην εγγραφή μετρήσεων όταν το 50% είναι αιτήματα για εγγραφή και το υπόλοιπο 50% είναι αιτήματα για ανάγνωση όπως και ο χρόνος που χρειάζεται για μια μέτρηση να παρουσιαστεί. Για την MySql ο αριθμός αιτημάτων που μπορεί να ολοκληρώνει το δευτερόλεπτο είναι το ένα τρίτο από την βάση δεδομένων Cassandra (4169 με 13082) και ο αριθμός εισαγωγής μιας μέτρησης είναι το διπλάσιο από οποιαδήποτε NoSql βάση δεδομένων (301 ms με 121 ms). Το γεγονός αυτό δικαιολογείται αφού είναι σχεσιακή βάση και πρέπει να επιβάλλει κανόνες ACID και να τηρεί τους κανόνες συσχέτισης του μοντέλου δεδομένου της .

Η MongoDb έχει αρκετά καλή απόδοση στην εγγραφή δεδομένων στην βάση αλλά όταν υπάρχουν αιτήματα ανάγνωσης δεν μπορεί να τα διαχειριστεί αρκετά καλά και φτάνει στους χρόνους της MySql . Αυτό οφείλεται κυρίως στο ότι ο προτεινόμενος αριθμός μηχανών για εγκατάσταση cluster με MongoDb είναι εννιά και στο παρόν πείραμα χρησιμοποιήθηκαν τέσσερις . Καταλαβαίνουμε ότι η MongoDb έχει μεγάλη ανάγκη από πόρους για να μπορεί να αποδώσει σωστά . Η MongoDb έχει κυρίως ανάγκη σε μνήμη τυχαίας προσπέλασης και υπολογιστική ισχύ στα shards αφού παρατηρήθηκε ότι ο επεξεργαστής βρισκόταν συνεχώς στο 100% και η μνήμη τυχαίας προσπέλασης ήταν σχεδόν γεμάτη. Ο configuration και router server είχαν λίγο πιο χαμηλά ποσοστά.

Η CouchBase και CassandraDb έχουν τα καλύτερα αποτελέσματα σχεδόν σε όλα τα πειράματα που εκτελέσθηκαν . Σε σχέση με την MySql έχουν τον διπλάσιο αριθμό

ολοκλήρωσης αιτημάτων το δευτερόλεπτο όπου η Couchbase έχει 13082 και η MySql 4062. Χρειάζονται τον μισό χρόνο εισαγωγής μια εγγραφής σε σχέση με την MySql και ο χρόνος για από την αποστολή μέχρι την παρουσίαση μιας εγγραφής είναι πέντε φορές πιο γρήγορος από την MySql. Όταν υπάρχει συνδυασμός αιτήσεων εγγραφής και ανάγνωσης δεδομένων υπάρχει ακόμα μεγαλύτερη διαφορά, δέκα φορές πιο γρήγορες από MySql και σε αυτή την περίπτωση έντεκα φορές πιο γρήγορες από MongoDb που δεν αποδίδει όσο καλά θα περιμέναμε για τους λόγους που αναφέραμε. Επίσης ο αριθμός ανοικτών συνδέσεων που μπορούν να διαχειριστούν είναι ο τριπλάσιος από MySql. Όταν υπάρχει μόνο ανάγνωση δεδομένων δεν είναι τόσο μεγάλη η διαφορά όταν ο αριθμός των υπάρχων εγγραφών που είναι αποθηκευμένος είναι μικρός. Όταν όμως αυξάνονται όπως είδαμε στο πείραμα η CassandraDb και Couchbase έχουν τρεις φορές καλύτερη απόδοση . Στο πείραμα όπου οι αισθητήρες στέλνουν με τυχαία περίοδο δεδομένα βλέπουμε μια πολύ σημαντική διαφορά στην εισαγωγή δεδομένων σε σχέση με Couchbase και MySql όπου η Couchbase είναι μέχρι οκτώ φορές πιο γρήγορη .

Οι διαφορές μεταξύ CassandraDb και Couchbase είναι μικρές μπορούν όμως να κάνουν την διαφορά σε κάποιες περιπτώσεις . Η CassandraDb μπορεί να διαχειριστεί περισσότερες συνδέσεις (διαφορά 3000 ανοικτές συνδέσεις) , αυτό όμως κοστίζει στον χρόνο δημιουργίας της πρώτης σύνδεσης . Απαντά σε περισσότερο αριθμό από αιτήματα το δευτερόλεπτο (διαφορά 3209 αιτήματα το δευτερόλεπτο) και έχει την καλύτερη απόδοση όταν τα αιτήματα είναι ανάγνωσης και εγγραφής . Από την άλλη η Couchbase έχει καλύτερη επίδοση στην εγγραφή δεδομένων (κατά μέσο όρο 30 ms), στο χρόνο παρουσίασης τους (κατά μέσο όρο 24 ms), και όταν τα δεδομένα στέλνονται με τυχαία περίοδο χρειάζεται λιγότερος χρόνος εγγραφής . Η CassandraDb όταν υπάρχει ανάγνωση δεδομένων είναι πιο αποδοτική με διαφορά κατά μέσο όρο 33 ms όταν αυξάνεται ο αριθμός των δεδομένων που διαβάζονται και και 31 ms όταν αυξάνεται ο αριθμός των αποθηκευμένων δεδομένων.

6.3 Γενικά συμπεράσματα

Στην παρούσα διπλωματική εργασία υλοποιήθηκε ένα μοντέλο δεδομένων για τις ανάγκες του Internet of Things το οποίο εφαρμόστηκε στις τέσσερις ακόλουθες βάσεις : MySql, CassandraDb, MongoDb και Couchbase . Για σκοπούς εξέτασης του μοντέλου δεδομένων και των βάσεων υλοποιήθηκε μια διεπαφή προγραμματισμού εφαρμογών για εισαγωγή των δεδομένων στην βάση, restful service για διαχείριση των αιτημάτων και πρόγραμμα παραγωγής δεδομένων ώστε να στέλνει τα δεδομένα στην βάση .

Όπως παρατηρήθηκε από τα αποτελέσματα όλες οι βάσεις έχουν τα δυνατά σημεία και τις αδυναμίες τους . Το μοντέλο δεδομένων δουλεύει αρκετά καλά σε όλες τις βάσεις αλλά για το Internet of things ιδανικές είναι οι NoSql βάσεις . Από τις NoSql βάσεις η CassandraDb και Couchbase είχαν τα καλύτερα αποτελέσματα και ήταν σταθερές σε όλα τα πειράματα με την απόδοση και των δύο βάσεων να βρίσκεται σε πολύ κοντινά επίπεδα . Οι δύο βάσεις έχουν αρκετά ψηλό αριθμό απαντήσεων αιτημάτων το δευτερόλεπτο και η εισαγωγή δεδομένων στις βάσεις γίνεται αρκετά πιο γρήγορα από την MongoDb και τη MySql . Η διαχείριση αιτημάτων ανάγνωσης και εισαγωγής δεδομένων γίνεται χωρίς καμιά δυσκολία σε αντίθεση με τις υπόλοιπες βάσεις . Το πιο σημαντικό είναι ότι ο χρόνος που χρειάζεται από την αποστολή μιας μετρικής μέχρι την εμφάνιση της είναι αρκετά λίγος και όσο αυξάνονται οι μετρικές που στέλνουν δεδομένα ο χρόνος που απαιτείται δεν φαίνεται να αυξάνεται σημαντικά . Όπως έχει ήδη αναφερθεί ένα σημαντικό πρόβλημα που παρουσιάζει η CouchBase είναι ότι δεν υποστηρίζει την νέα έκδοση ubuntu 16.04 και το γεγονός αυτό έγινε αιτία αρκετών προβλημάτων . Η CassandraDb ήταν η πιο ευκολόχρηστη βάση στη διεξαγωγή της πειραματικής μας διαδικασίας αφού η εγκατάστασή της στις μηχανές ήταν εύκολη όπως επίσης και η διαχείρισή της .

Βιβλιογραφία

[1] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems, 29*(7), 1645-1660. doi:10.1016/j.future.2013.01.010

[2] Internet of Things: A survey - miun.se. (n.d.). Retrieved May 8, 2017, from https://www.bing.com/cr?IG=B9DC9F39B908451DA6FF93FD20A4D50A&CID=378C7C14405E6BAC0544766841CE6AB6&rd=1&h=ddGWT-id1rfRxNdCSZ9wxBlSsfsjne4q9y9U3KnCfWA&v=1&r=https%3a%2f%2fwww.miun.se%2fsiteassets%2ffakulteter%2fnmt%2fsummer-university%2f1-s20-s1389128610001568-mainpdf&p=DevEx,5054.1

[3] The Cloud is Not Enough: Saving IoT from the Cloud. (n.d.). Retrieved May 8, 2017, from https://www.bing.com/cr?IG=FD1CFED98848459E8FD24D24FA1B6C01&CID=1A115BE9D8E2682D12F45195D97269EE&rd=1&h=bJ1kA8kbzO32q1RoPzqBygPdWeHe5dno19j-xYMyEBk&v=1&r=https%3a%2f%2fwww.usenix.org%2fsystem%2ffiles%2fconference%2fhotcloud15%2fhotcloud15-zhang.pdf&p=DevEx,5062.1

[4] Khan, R., Khan, S. U., Zaheer, R., & Khan, S. (2012). Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. *2012 10th International Conference on Frontiers of Information Technology*. doi:10.1109/fit.2012.53

[5] Thihinas,D.,Pallis,G.,Dikaiakos,M.(2016), *Low Cost Adaptive Monitoring Techniques for the Internet of Things*.IEEE Transactions on big data.

[6] Cao, T., Pham, T., Vu, Q., Truong, H., Le, D., & Dustdar, S. (2016). Marsa. *MARSA: A Marketplace for Realtime Human Sensing Data*, 16(3), 1-21. doi:10.1145/2883611

[7] Spatio-Temporal Data Types: An Approach to Modeling and ... (n.d.). Retrieved May 8, 2017, from http://www.bing.com/cr?IG=0B1965A3592B4EB8A316DEA45E3BDF04&CID=1EA063B5699F6E09136E69C9680F6F41&rd=1&h=g1xWyLIOuum1N1UGKIBAJUxk6fm8GbpNoQmm-_ZXAig&v=1&r=http%3a%2f%2feecs.oregonstate.edu%2f%257Eerwig%2fpapers%2fMovingObjects_GEOINF99.pdf&p=DevEx,5062.1

[8] Abu-Elkheir, M., Hayajneh, M., & Ali, N. (2013). Data Management for the Internet of Things: Design Primitives and Solution. *Sensors, 13*(11), 15582-15612. doi:10.3390/s131115582

[9] Miorandi, D., Sicari, S., Pellegrini, F. D., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. *Ad Hoc Networks, 10*(7), 1497-1516. doi:10.1016/j.adhoc.2012.02.016

[10] Pujolle, G. (2006). An Autonomic-oriented Architecture for the Internet of Things. IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06). doi:10.1109/jva.2006.6

[11] Cassandra - A Decentralized Structured Storage System. (n.d.). Retrieved May 21, 2017, from https://www.bing.com/cr?IG=958C1FFD76A241648F3FA9D91119AFF0&CID=12E29592D69E684E1D289F1BD79869F7&rd=1&h=50eM-CrgSYEFy5LvhVlZAuLXeoqR9j0tF-662LOlxYw&v=1&r=https%3a%2f%2fwww.cs.cornell.edu%2fprojects%2fladis2009%2fpapers%2flakshman-ladis2009.pdf&p=DevEx,5058.1

[12] MongoDB vs Oracle - database comparison - ResearchGate. (n.d.). Retrieved May 21, 2017, from https://www.bing.com/cr?IG=50C68C5934EF411D8C56EF285F32ED13&CID=24C2CD96F9B469221EA9C71FF8B268E3&rd=1&h=E-ClpTeIWU7z93xCbqhOLJawye4zr5KWi4kWsHM7HpE&v=1&r=https%3a%2f%2fwww.researchgate.net%2fprofile%2fAlexandru_Boicea%2fpublication%2f261040647_MongoDB_vs_Oracle_--_Database_Comparison%2flinks%2f55c2132b08aebc967defd053.pdf&p=DevEx,5099.1

[13] Jing, Q., Vasilakos, A. V., Wan, J., Lu, J., & Qiu, D. (2014). Security of the Internet of Things: perspectives and challenges. *Wireless Networks, 20*(8), 2481-2501. doi:10.1007/s11276-014-0761-7

[14] Oracle MySql, https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html

[15] Apache Cassandra, http://cassandra.apache.org/doc/latest/cql/index.html

[16] MongoDb, https://docs.mongodb.com/manual/

[17]CouchbBase, https://developer.couchbase.com/documentation/server/4.6/introduction/intro.html

Παράρτημα Α

Ερωτήματα και παραδείγματα εκτέλεσης

Στην διεπαφή IDbHandler υπάρχουν τα πιο κάτω ερωτήματα (queries) τα οποία υλοποιούνται για τις βάσεις MySql, Cassandra, MongoDb και CouchBase. Ακολουθούν παραδείγματα των ερωτημάτων και η έξοδος τους.

Επιλογή όλων των application:
SELECT * FROM Applications;
Output: '0c60a2dd-5062-4613-9ffa-4d98ef4f8e41', 'app8', 'desc'

Επιλογή συγκεκριμένου application.
SELECT * FROM Applications
WHERE appId = '17e0f2a5-d025-49f1-90c8-a66eebed0131';
Output: '17e0f2a5-d025-49f1-90c8-a66eebed0131', 'app4', 'desc'

Επιλογή όλων των sensors.
SELECT * FROM Sensors;
Output:
'0bba1738-dce3-4166-b531-8a444ab4170f', '00c99a8f-be9a-461d-a81d-d41f7707ae9c', 'app3_sensor5', 'desc'

Επιλογή συγκεκριμένου sensor.
SELECT * FROM Sensors
WHERE sensorId = '011fa6bf-bec9-40e2-8def-63f111d93230';
Output: '17e0f2a5-d025-49f1-90c8-a66eebed0131', '011fa6bf-bec9-40e2-8def-63f111d93230', 'app4_sensor4', 'desc'

Επιλογή συγκεκριμένου μετρικών κάποιου αισθητήρα.

```
SELECT * FROM Metrics
WHERE sensorId = ''011fa6bf-bec9-40e2-8def-63f111d93230'';
```

Output:

'17e0f2a5-d025-49f1-90c8-a66eebed0131', '011fa6bf-bec9-40e2-8def-63f111d93230', '011fa6bf-bec9-40e2-8def-63f111d93230$jxbipkhzhqhacpzbr', 'Text', 'jxbipkhzhqhacpzbr'

Επιλογή όλων των μετρήσεων για συγκεκριμένες ημερομηνίες και ώρες συγκεκριμένης μέτρησης.

```
SELECT * FROM Measurements WHERE (timestamp BETWEEN '1485361912' AND '1485361913')
AND (metricId = '00c99a8f-be9a-461d-a81d-d41f7707ae9c$aygndfhirpshhanisix');
```

Output:

'00c99a8f-be9a-461d-a81d-d41f7707ae9c$aygndfhirpshhanisix', 'wruvxwswcqtaslvc', '1485361912'