ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
Πρόγραμμα Μεταπτυχιακών Σπουδών
"Πληροφοριακά Συστήματα & Υπηρεσίες"

UNIVERSITY OF PIRAEUS
DEPARTMENT OF DIGITAL SYSTEMS
Postgraduate Programme
"Information Systems & Services"

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφοριακά Συστήματα & Υπηρεσίες»

Ειδίκευση: *Μεγάλα Δεδομένα και Αναλυτική*

Διαχείριση Δεδομένων για Σχεσιακές και μη Σχεσιακές Βάσεις Δεδομένων

Διδάσκων:

Επίκουρος Καθηγητής

Χρήστος Δουλκερίδης

Φοιτητές:

Κωνσταντίνος Πλατής ΜΕ1940

Χρήστος Καπότης ΜΕ 1926

Αθήνα 2020

Περίληψη

Σκοπός της εργασίας είναι η ανάπτυξη ενός συστήματος διαχείρισης των δεδομένων του συστήματος αυτόματης αναγνώρισης (AIS data), με χρήση του μη σχεσιακού συστήματος βάσης δεδομένων MongoDB. Αρχικά, γίνεται μια σύντομη αναφορά στο σύστημα αυτόματης αναγνώρισης και στη μεγάλη σημασία που διαδραματίζει στις μέρες μας στο χώρο της ναυσιπλοΐας. Εν συνεχεία, παρουσιάζεται η οργάνωση των δεδομένων μέσα στη βάση και μελετάται η αποτελεσματικότητα της άντλησης πληροφοριών μέσω χωρικών επερωτήσεων που λαμβάνουν χώρα. Μελετώνται διάφορα υποθετικά σενάρια, τα οποία θα μπορούσαν να αποτελέσουν πραγματικά χρήσιμα ερωτήματα στο χώρο της ναυσιπλοΐας. Τέλος, παρατίθεται μια εκτενής αναφορά στις τεχνικές επεξεργασίας και διαχείρισης των δεδομένων μέσα σε μία μή σχεσιακή βάση MongoDB (αποθήκευση, διαχείριση, replication, sharding).

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ 3
Κεφάλαιο 1^{ο} : Εισαγωγή 4
Κεφάλαιο 2^{ο} : AIS data 5
2.1 Γενικές πληροφορίες 5
2.2 Σύνολο δεδομένων προς μελέτη 6
Κεφάλαιο 3^{ο} : Η μη σχεσιακή βάση δεδομένων MongoDB 8
3.1 Μη σχεσιακές Βάσεις Δεδομένων 8
3.2 Κατηγορίες μη Σχεσιακών Βάσεων Δεδομένων 9
3.3 MongoDB 11
Κεφάλαιο 4^{ο}: Υλοποίηση-επεξεργασία δεδομένων 13
4.1 Προεπεξεργασία δεδομένων 13
4.2 Αποθήκευση δεδομένων 15
4.3 Use case 1: aggregation 15
4.4 Use case 2: indexing 17
Κεφάλαιο 5^{ο} : Replication-Sharding 20
5.1 Replication 20
5.2 Εφαρμογή replication 21
5.3 Sharding 22
Κεφάλαιο 6^{ο}: Συμπεράσματα 25
Βιβλιογραφία 26

Κεφάλαιο 1^{ο} : Εισαγωγή

Το σύστημα αυτόματης αναγνώρισης (AIS) επιτρέπει την ταυτοποίηση, τον εντοπισμό και την παρακολούθηση των σκαφών παγκοσμίως. Είναι το πιο συχνά χρησιμοποιούμενο θαλάσσιο σύστημα σε όλο τον κόσμο. Παρέχει πλούσια δεδομένα κίνησης σκαφών σε πραγματικό χρόνο και πληροφορίες για τον εντοπισμό άλλων σκαφών, καθώς επίσης και πληροφορίες, όπως το όνομα, την ταχύτητα του σκάφους, τον προορισμό κ.α. Το AIS χρησιμοποιείται ευρέως σε όλο τον κόσμο για να αποφευχθούν συγκρούσεις σκαφών, για ασφάλεια, για αποδοτικότητα της ναυσιπλοΐας, για την επικοινωνία μεταξύ σκαφών και μεταξύ πλοίων και ακτών. Το AIS χρησιμοποιείται κυρίως από την υπηρεσία κυκλοφορίας σκαφών (VTS) λόγω της παροχής θέσεων σε πραγματικό χρόνο και άλλων ενδιαφερόντων πληροφοριών για τα γειτονικά σκάφη. Γίνεται αντιληπτό, λοιπόν, ότι η διαδικασία αποθήκευσης, επεξεργασίας και άντλησης πληροφοριών από αυτό, είναι μείζονος σημασίας. Για το λόγο αυτό, στην παρούσα εργασία μελετάται ο τρόπος διαχείρισης των δεδομένων μέσα από τη μή σχεσιακή βάση δεδομένων mongoDB.

Όσον αφορά τη δομή της εργασίας , στο δεύτερο κεφάλαιο γίνεται μια αναφορά στο σύστημα αυτόματης αναγνώρισης και στον τρόπο λειτουργίας αυτού. Μελετώνται τα χαρακτηριστικά, το είδος και ο τρόπος συγκέντρωσης των δεδομένων. Στο τρίτο κεφάλαιο, παρουσιάζεται το σύστημα διαχείρισης δεδομένων σε μή σχεσιακές βάσεις δεδομένων και συγκεκριμένα στη MongoDB, η οποία και επιλέχθηκε για την ανάλυση μας στην παρούσα εργασία. Ακολουθεί, στο τέταρτο κεφάλαιο, η υλοποίηση και επεξεργασία των δεδομένων με σκοπό την άντληση χρήσιμων πληροφοριών από αυτά. Στο πέμπτο κεφάλαιο παρουσιάζονται οι διαδικασίες του replication και sharding. Τέλος, στο έκτο κεφάλαιο, παρατίθενται τα συμπεράσματα στα οποία καταλήξαμε ύστερα από την εκπόνηση της εργασίας. Ακολουθεί η βιβλιογραφία.

Κεφάλαιο 2^{ο} : AIS data

2.1 Γενικές πληροφορίες

Τα AIS δεδομένα αποτελούνται από ψηφιακά μηνύματα που ανταλλάσουν τα σκάφη και οι σταθμοί μεταξύ τους.Τα σκάφη είναι σε θέση να στείλουν 27 διαφορετικά είδη μηνυμάτων AIS. Ορισμένα μηνύματα περιέχουν τη θέση ή τη ταχύτητα του σκάφους (δυναμικές πληροφορίες), ενώ άλλα περιέχουν «στατικές» πληροφορίες σχετικά με τη διαδρομή , όπως το όνομα και τον τύπο του σκάφους, τις διαστάσεις και τον προορισμό του (στατικές πληροφορίες). Δεν υπάρχει ένα μοναδικό μήνυμα που να περιέχει όλες τις διαθέσιμες πληροφορίες ενός σκάφους. Τα δεδομένα που περιέχουν αυτά τα μηνύματα AIS καταγράφονται είτε από αισθητήρες, είτε από το παγκόσμιο σύστημα εντοπισμού θέσης (GPS), είτε, τέλος, παρέχονται χειροκίνητα από τον ιδιοκτήτη του εκάστοτε σκάφους. Όλα τα θαλάσσια σκάφη άνω των 300 ακαθάριστων τόνων , τα εμπορικά και τα σκάφη μακρύτερα των 20 μέτρων υποχρεούνται σύμφωνα με στον Διεθνή Ναυτιλιακό Οργανισμό να διαθέτουν σύστημα AIS επί του σκάφους, το οποίο να αποτελείται από αισθητήρες και αναμεταδότες.

Ο ρυθμός με τον οποίο τα σκάφη στέλνουν τα μηνύματά τους εξαρτάται από τον τύπο του μηνύματος. Ο ρυθμός ενημέρωσης των μηνυμάτων AIS που περιέχουν τη θέση των σκαφών εξαρτάται από την ταχύτητα του σκάφους. Η συχνότερη ενημέρωσης για αυτά τα μηνύματα είναι μία φορά κάθε δύο δευτερόλεπτα. Για τα μηνύματα που περιέχουν στατικές πληροφορίες, όπως το όνομα του πλοίου ή ο προορισμός, ο ρυθμός ενημέρωσης είναι πολύ μικρότερος, μία φορά κάθε 3 έως 6 λεπτά, ανάλογα με το αν το σκάφος είναι εν κινήσει.

Όπως αναφέρθηκε και παραπάνω, τα δεδομένα AIS χρησιμοποιούνται κατα κόρον από την υπηρεσία κυκλοφορίας σκαφών λόγω της παροχής θέσεων σε πραγματικό χρόνο. Ωστόσο, η ανάλυση των δεδομένων και σε μεταγενέστερο χρόνο μπορεί να αναδείξει σημαντικές πληροφορίες για διάφορες άλλες εφαρμογές. Κατά συνέπεια, κρίνεται επιτακτική η αποθήκευση των πληροφοριών αυτών σε βάσεις δεδομένων, έτσι ώστε να καθίσταται ευκολότερη και αμεσότερη η πρόσβαση σε αυτές. Για παράδειγμα μέσα από τη μελέτη και επεξεργασία των δεδομένων μπορούν εύκολα να εντοπισθούν τα σκάφη που παραβιάζουν τα πρωτόκολλα ναυσιπλοΐας, όπως τα

όρια ταχύτητας κατά την είσοδο τους σε λιμάνια ή ναύσταθμους. Άλλο ένα ενδιαφέρον ερώτημα που θα μπορούσε να απαντηθεί από τη μελέτη των δεδομένων AIS θα μπορούσε να είναι η εύρεση των πλοίων (όνομα, τύπος, προορισμός και ταχύτητα πλοίου) σε συγκεκριμένα σημεία του χάρτη (lon, lat). Κατά αυτόν τον τρόπο, καθίσταται δυνατό να γνωρίζουμε κάθε συγκεκριμένη χρονική στιγμή, ποιό σκάφος ήταν σε ένα συγκεκριμένο σημείο. (πχ. εντοπισμός πλοίου που άφησε λήμματα σε προστατευμένες ζώνες).

2.2 Σύνολο δεδομένων προς μελέτη

Πηγή πληροφοριών για τη διενέργεια της μελέτης αποτέλεσε το ετερογενές ολοκληρωμένο σύνολο δεδομένων για τη ναυτική ευφυία, επιτήρηση και αναγνώριση (Heterogeneous Integrated Dataset for Maritime Intelligence, Surveillance, and Reconnaissance) που διατίθεται ελεύθερα στην ιστοσελίδα https://zenodo.org/record/1167595 . Το σύνολο δεδομένων περιέχει τέσσερις κατηγορίες στοιχείων:

- Δεδομένα πλοήγησης
- Δεδομένα αναφερόμενα σε πληροφορίες σκαφών
- Γεωγραφικά δεδομένα
- Περιβαλλοντικά δεδομένα

Καλύπτει χρονικό διάστημα έξι μηνών, από την 1η Οκτωβρίου 2015 έως την 31η Μαρτίου 2016 και παρέχει θέσεις πλοίων στην Κελτική Θάλασσα, τη Μάγχη και το Βισκαϊκό κόλπο (Γαλλία).

Η κατηγορία δεδομένων που χρησιμοποιήθηκε για την ανάλυση ήταν τα δεδομένα πλοήγησης AIS Data. Απαρτίζονται από τρία αρχεία csv με τίτλους nari.dynamic, nari.dynamic.sar και nari.static. Οι πληροφορίες που περιέχουν τα προαναφερόμενα αρχεία παρουσιάζονται παρακάτω.

- Nari_dynamic: [, sourcemmsi, speedoverground, lon, lat]
- Nari_dynamic_sar: [, sourcemmsi, speedoverground, ts]
- Nari_static: [, sourcemmsi, shipname, shiptype, tobow, tostern, destination]

όπου sourcemmsi: αναγνωριστικό σκάφους (μοναδικό για κάθε σκάφος), speedovergound: ταχύτητα σκάφους σε κόμβους, lon: γεωγραφικό μήκος σε μοίρες, lat: γεωγραφικό πλάτος σε μοίρες, ts: χρονολογία σε unix epochs, shipname: όνομα σκάφους, shiptype: τύπος σκάφους, tobow: απόσταση γέφυρας-πλώρης σε μέτρα, tostern: απόσταση γέφυρας-πρύμνης σε μέτρα, destination: προορισμός. Η πρώτη στήλη που δεν διαθέτει ονομασία (“ ”) αφορά στη θέση της εγγαφής μέσα στο αρχείο.

Όπως γίνεται κατανοητό, επειδή τα δεδομένα μεταβάλλονται συνεχώς κρίνεται πιο εξυπηρετικό, η αποθήκευση τους να πραγματωθεί όχι σε ένα παραδοσιακό σχεσιακό σύστημα αποθήκευσης αλλά σε μία μή σχεσιακή βάση δεδομένων. Για την παρούσα εργασία επιλέχθηκε το σύστημα αποθήκευσης της MongoDB.

Κεφάλαιο 3^{ο} : Η μη σχεσιακή βάση δεδομένων MongoDB

3.1 Μη σχεσιακές Βάσεις Δεδομένων

Η προσέγγιση NoSQL (συντομογραφία του not only SQL) αναφέρεται σε μια τάξη συστημάτων διαχείρισης βάσεων δεδομένων τα οποία δεν χρησιμοποιούν την SQL ως γλώσσα ερωτημάτων και δεν ακολουθούν τους κανόνες της σχεσιακής σχεδίασης. Τα συστήματα αυτά είναι δομημένα με διαφορετικό τρόπο σε σύγκριση με τις σχεσιακές βάσεις δεδομένων. Οι NoSQL βάσεις χρησιμοποιούνται σε περιβάλλοντα στα οποία ο όγκος δεδομένων είναι πολύ μεγάλος και παρουσιάζονται προβλήματα απόδοσης λόγω των εγγενών περιορισμών της SQL. Τα εν λόγω συστήματα δεν συμμορφώνονται πιστά στους κανόνες ACID (Atomicity, Consistency, Integrity, Durability) των σχεσιακών βάσεων, έτσι ώστε να ανταποκρίνονται σε συνθήκες στις οποίες η απόδοση ενός ερωτήματος είναι πιο σημαντική από την απόλυτη συνέπεια των δεδομένων. Οι βάσεις NoSQL θυσιάζουν τις αυστηρές απαιτήσεις σε συνέπεια για υψηλές ταχύτητες και ελαστικότητα. Επιπλέον, σε αντίθεση με τις σχεσιακές βάσεις που είναι αυστηρά δομημένες, τα δεδομένα στις NoSQL δεν περιορίζονται εν γένει από κάποιο σχήμα. Ένα από τα σημαντικότερα χαρακτηριστικά τους είναι ότι παρέχουν υψηλή διαθεσιμότητα στα δεδομένα του συστήματος. Η φιλοσοφία των NoSQL εστιάζει στα κατανεμημένα συστήματα βάσεων, όπου αδόμητα δεδομένα αποθηκεύονται σε πολλαπλούς κόμβους. Αυτή η κατανεμημένη αρχιτεκτονική επιτρέπει την οριζόντια κλιμάκωση του συστήματος, δίνοντας την δυνατότητα να προστίθενται συνεχώς νέοι πόροι καθώς τα δεδομένα αυξάνονται, χωρίς επιβάρυνση στην απόδοση.

Σύμφωνα με το θεώρημα CAP (Eric Brewer, University of California, Berkeley) υπάρχουν τρία επιθυμητά χαρακτηριστικά για κάθε σύστημα δεδομένων που έχει αναπτυχθεί σε κατανεμημένο περιβάλλον.

- Συνέπεια (Consistency) : κάθε κόμβος στο σύστημα περιέχει τα ίδια δεδομένα.
- Διαθεσιμότητα (Availability): κάθε αίτηση σε ένα λειτουργικό κόμβο στο σύστημα επιστρέφει μια απάντηση.

- Ανοχή στο διαχωρισμό (Partition tolerance) : οι ιδιότητες του συστήματος (η συνέπεια και η διαθεσιμότητα) έχουν ισχύ ακόμη και αν μεμονωμένα συστατικά του δεν είναι διαθέσιμα.

Το θεώρημα αναφέρει ότι κάθε κατανεμημένο σύστημα μπορεί να υποστηρίξει, το πολύ, δύο από αυτές τις ιδιότητες.

3.2 Κατηγορίες μη Σχεσιακών Βάσεων Δεδομένων

Υπάρχουν τέσσερις τύποι βάσεων δεδομένων NoSQL: βάσεις κλειδιού-τιμής (key-value), βάσεις αποθήκευσης κατά στήλες (columnar systems), βάσεις εγγράφων (Document Databases) και βάσεις γράφων (Graph Database). Αυτοί οι τέσσερις τύποι βάσεων δεδομένων είναι διαφορετικοί στις λειτουργίες τους και καθένας εξ αυτών ειδικεύεται σε ορισμένες περιπτώσεις χρήσης.

I. Βάσεις κλειδιού-τιμής (key-value). Χρησιμοποιείται ένας πίνακας κατακερματισμού με ένα μοναδικό κλειδί και ένα δείκτη στο αντίστοιχο στοιχείο δεδομένων. Οι βάσεις κλειδί-τιμή δεν απαιτούν σχήμα και προσφέρουν μεγάλη ευελιξία και επεκτασιμότητα ,δεν προσφέρουν τη δυνατότητα ατομικότητας, συνεκτικότητας, απομόνωσης, διατηρησιμότητα (ACID) και απαιτούν κάποια εργαλεία για την τοποθέτηση των δεδομένων, την αποφυγή διπλών αντιγράφων και την ανοχή σε σφάλματα καθώς όλα αυτά δεν ελέγχονται ρητά από την ίδια την τεχνολογία. Προτιμάται για δεδομένα απλής φύσης, που μπορούν να αναπαρασταθούν με τη μορφή κλειδιού – τιμής (session management, shopping cart κλπ). Οι πιο γνωστές είναι η Voldemort, Redis.
II. Βάσεις αποθήκευσης κατά στήλες (columnar systems) Χρησιμοποιούνται για την αποθήκευση και επεξεργασία πολύ μεγάλων ποσοτήτων δεδομένων κατανεμημένων σε πολλά μηχανήματα. Οι σχεσιακές βάσεις είναι προσανατολισμένες κατά γραμμή καθώς τα δεδομένα σε κάθε γραμμή του πίνακα 13 αποθηκεύονται μαζί. Σε κατά στήλη προσανατολισμένες βάσεις τα δεδομένα αποθηκεύονται κατά μήκος των γραμμών. Είναι πολύ εύκολο να προσθέσουμε στήλες και μπορούν να προστεθούν σειρά-σειρά προσφέροντας μεγαλύτερη ευελιξία, απόδοση και επεκτασιμότητα. Όταν

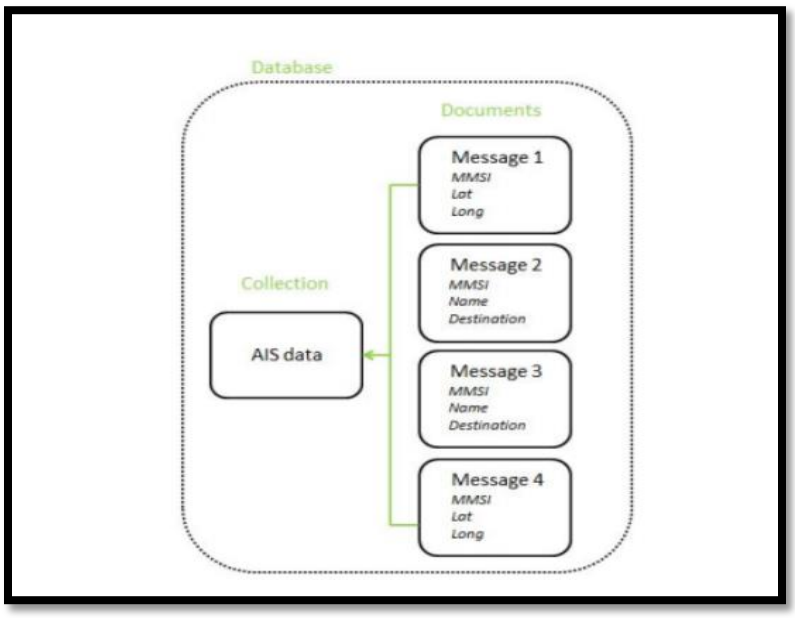
υπάρχει μεγάλος όγκος και ποικιλία δεδομένων είναι μια πολύ καλή λύση. Είναι ευπροσάρμοστη αφού το μόνο που κάνουμε είναι η προσθήκη κι άλλων στηλών. Προτιμάται για πίνακες δεδομένων, πολύ μεγάλου μεγέθους, στους οποίους συνήθως ανατίθενται ερωτήματα που αφορούν μόνο λίγες από τις διαθέσιμες στήλες (αποθήκες δεδομένων, CRM, κλπ). Οι πιο γνωστές είναι οι BigTable, Hbase, Hypertable, Cassandra.

III. Βάσεις εγγράφων (Document Databases) Είναι παρόμοιες με τις key-value αλλά βασίζονται σε έγγραφα που αποτελούν συλλογές από άλλες key-value συλλογές, υπάρχει δηλαδή εμφώλευση. Μία βάση εγγράφων απαιτεί από τα δεδομένα να αποθηκεύονται σε μια συγκεκριμένη μορφή, η οποία θα είναι κατανοητή από τη βάση. Η μορφή μπορεί να είναι XML, JSON, Binary JSON (BSON), ή οτιδήποτε άλλο η βάση δεδομένων μπορεί να αντιληφθεί. Τέτοιες βάσεις είναι συνήθως χρήσιμες όταν υπάρχουν αρκετές αναφορές κι αυτές παράγονται και συναρμολογούνται από στοιχεία που αλλάζουν συχνά. Προτιμάται για δεδομένα που ακολουθούν τη μορφή εγγράφων (παραστατικά, συνταγές φαρμάκων, παραπεμπτικά κλπ). Οι πιο διαδεδομένες αυτής της κατηγορίας είναι η MongoDB, η Elasticsearch και η CouchDB.

IV. Βάσεις γράφων (Graph Database) Η βασική δομή αποκαλείται και “σχέση κόμβων”. Η δομή αυτή είναι χρήσιμη όταν έχουμε να κάνουμε με διασυνδεδεμένα δεδομένα. Οι κόμβοι και οι σχέσεις υποστηρίζουν κάποιες ιδιότητες, δηλαδή ένα ζεύγος key-value όπου αποθηκεύονται τα δεδομένα. Η πλοήγηση στη βάση γίνεται ακολουθώντας τις σχέσεις. Αυτού του είδους η αποθήκευση και πλοήγηση στα συστήματα RDBMS δεν είναι δυνατή εξαιτίας της ακαμψίας της δομής των πινάκων και της αδυναμίας να ακολουθηθούν οι συνδέσεις μεταξύ των στοιχείων, όπου κι αν αυτές οδηγούν. Προτιμάται για δεδομένα που έχουν μεγάλη συσχέτιση μεταξύ τους και η συσχέτιση αυτή πρέπει να αποτυπωθεί (social networks). Neo4j, FlockDB, OrientDB, AllegroGraph, GraphDB Παραδείγματα τέτοιων βάσεων είναι το Neo4J, το FlockDB, το OrientDB, το AllegroGraph και το GraphDB.

3.3 MongoDB

Το σύστημα βάσης δεδομένων που επιλέχθηκε για την εκπόνηση της εργασίας είναι η MongoDB. Αποτελεί ένα διαπλατφορμικό σύστημα βάσης δεδομένων που βασίζεται σε έγγραφα (cross-platform document-oriented database system) και κατηγοριοποιείται ως NoSQL βάση δεδομένων. Η MongoDB δεν έχει την παραδοσιακή δομή μίας σχεσιακής βάσης δεδομένων με πίνακες, αλλά χρησιμοποιεί JSON-like έγγραφα με δυναμικά schemas, καθιστώντας την ενσωμάτωση των δεδομένων σε ορισμένους τύπους εφαρμογών ευκολότερη και ταχύτερη. Επίσης, το μοντέλο αποθήκευσης σε έγγραφα, δίνει τη δυνατότητα αποθήκευσης των δεδομένων που αφορούν κάποια οντότητα συγκεντρωμένα σε ένα έγγραφο, χωρίς να χρειάζεται JOIN για την ανάκτησή τους (μείωση της κανονικοποίησης). Είναι γραμμένη σε C++ και είναι σχεδιασμένη να προσφέρει υψηλή απόδοση στις εφαρμογές, επεκτασιμότητα, υψηλή διαθεσιμότητα και δυνατότητα υποβολής σύνθετων ερωτημάτων. Για τα δεδομένα της εγγυάται τελική συνέπεια (eventual consistency). Εχει υιοθετήσει αρκετα θετικά χαρακτηριστικά των σχεσιακών συστημάτων δεδομένων όπως τα ευρετήρια (indexes) και τα σύνθετα ερωτήματα (complex queries). Όλα τα παραπάνω την καθιστούν το δημοφιλέστερο NoSQL σύστημα βάσεων δεδομένων.



Σχήμα 1. Ταξινόμηση δεδομένων στη MongoDB

Αξίζει, τέλος, να σημειωθεί ότι παρέχει τη δυνατότητα οριζόντιας κλιμάκωσης με ισοκατανομή των δεδομένων στους κόμβους (sharding) και αυτόματης αντιγραφής των δεδομένων (replication).

Κεφάλαιο 4ο: Υλοποίηση-επεξεργασία δεδομένων

4.1 Προεπεξεργασία δεδομένων

Πρωταρχικό μέλημα για την έναρξη της μελέτης αποτέλεσε ο καθορισμός του μεγέθους του συνόλου δεδομένων. Όπως γίνεται κατανοητό, τα αρχεία CSV που διατέθηκαν προς ανάλυση ήταν εξαιρετικά μεγάλα σε όγκο πληροφορίας ενώ περιείχαν και αρκετές ελλειπούσες τιμές. Αποφασίστηκε, λοιπόν, να προχωρήσουμε σε «καθαρισμό» των δεδομένων και συγκράτηση ενός μέρους του συνόλου. Κατά αυτόν τον τρόπο, περιορίστηκε σημαντικά και ο χρόνος εκτέλεσης των διαδικασιών-εντολών, από τη στιγμή που αποφασίστηκε να εργαστούμε τοπικά (εγκατάσταση του συστήματος MongoDB στον υπολογιστή μας). Συνολικά συγκρατήσαμε 10000, 4566 και 660423 εγγραφές από τα αρχεία “nari_dynamic”, “nari_dynamic_sar“ και “ nari_static” αντίστοιχα. Τα νέα καθαρισμένα, πλέον αρχεία ονομάστηκαν “aisdata_nari_dynamic_cleaned”, “aisdata_nari_dynamic_sar_cleaned” και “aisdata_nari_static_cleaned”. Επίσης αφαιρέθηκαν οι εγγραφές με κενές τιμές για καλύτερη επεξεργασία του συνόλου δεδομένων. Για τις παραπάνω διαδικασίες χρησιμοποιήθηκε η γλώσσα προγραμματισμού python, με τις κατάλληλες βιβλιοθήκες (pandas, numpy). Οι εντολές που χρησιμοποιήθηκαν παρατίθενται παρακάτω.

Nari_dynamic:

- *nari_dynamic = pd.read_csv('/home/christos/Master/Data Management in SQL and NoSQL DBs/DataSet/[P1] AIS Data/nari_dynamic.csv', low_memory=False, nrows=10000)*
- *nari_dynamic_cleaned = nari_dynamic[["sourcemmsi", "speedoverground", "lon", "lat"]]*
- *nari_dynamic_cleaned.to_csv(r'/home/christos/Master/Data Management in SQL and NoSQL DBs/DataSet/[P1] AIS Data/aisdata_nari_dynamic_cleaned.csv')*

Nari_dynamic_sar:

- *nari_dynamic_sar = pd.read_csv('/home/christos/Master/Data Management in SQL and NoSQL DBs/DataSet/[P1] AIS Data/nari_dynamic_sar.csv', low_memory=False, nrows= 4566)*
- *nari_dynamic_sar_new = nari_dynamic_sar[["sourcemmsi", "speedoverground", "ts"]]*
- *ts_toDateTime = pd.to_datetime(nari_dynamic_sar_new["ts"], unit='s')*
- *nari_dynamic_sar_new["ts"] = ts_toDateTime*
- *nari_dynamic_sar_cleaned = nari_dynamic_sar_new[["sourcemmsi", "speedoverground", "ts"]]*
- *nari_dynamic_sar_cleaned.to_csv(r'/home/christos/Master/Data Management in SQL and NoSQL DBs/DataSet/[P1] AIS Data/aisdata_nari_dynamic_sar_cleaned.csv')*

Nari_static:

- *nari_static = pd.read_csv('/home/christos/Master/Data Management in SQL and NoSQL DBs/DataSet/[P1] AIS Data/nari_static.csv', low_memory=False, nrows=660423)*
- *nari_static_new = nari_static[["sourcemmsi", "shipname", "shiptype", "tobow", "tostern", "destination"]]*
- *nari_static_new_drop = nari_static_new.dropna()*
- *filled = []*
- ```
  def cleaned(col):
      for c in col:
          if c != " ":
              filled.append(c)
          elif c == " ":
              c = np.nan
              filled.append(c)
  ```
- *cleaned(nari_static_new_drop["destination"])*
- *nari_static_new_drop['destination'] = filled*
- *nari_static_cleaned = nari_static_new_drop.dropna()*
- *nari_static_cleaned.to_csv(r'/home/christos/Master/Data Management in SQL and NoSQL DBs/DataSet/[P1] AIS Data/aisdata_nari_static_cleaned.csv')*
  ```

4.2 Αποθήκευση δεδομένων

Τα τρία «καθαρισμένα» αρχεία csv (nari_dynamic, nari_dynamic_sar και nari_static) φορτώθηκαν και αποθηκεύτηκαν στη σύστημα της βάσης δεδομένων μας από τη γραμμή εντολών της mongoDB. Το όνομα της βάσης ορίσθηκε ως «aisdata», με το ονόματα των συλλογών να είναι «nari_dynamic», «nari_dynamic_sar» και «nari_static» αντίστοιχα. Οι εντολές που χρησιμοποιήθηκαν παρουσιάζονται παρακάτω.

- *mongoimport --db aisdata --collection nari_dynamic --type csv --file C:\aisdata_nari_dynamic_cleaned.csv --headerline*
- *mongoimport --db aisdata --collection nari_dynamic_sar --type csv --file C:\aisdata_nari_dynamic_sar_cleaned.csv --headerline*
- *mongoimport --db aisdata --collection nari_static --type csv --file C:\aisdata_nari_static_cleaned.csv --headerline*

4.3 Use case 1: aggregation

Σαν πρώτη περίπτωση χρήσης του συνόλου δεδομένων θα μπορούσε να είναι η εύρεση της μέσης ταχύτητας των σκαφών, και εν συνεχεία η εύρεση των 2 σκαφών με τη μεγαλύτερη ταχύτητα. Έτσι, θα μπορούν να εντοπιστούν τα σκάφη που παραβιάζουν το πρωτόκολλα ναυσιπλοΐας ως προς τη μέγιστη ταχύτητα που οφείλουν να τηρούν, τόσο στην ανοιχτή θάλασσα, όσο κυρίως κατά την προσέγγιση τους σε λιμάνια. Όπως είναι γνωστό, λόγω κυκλοφοριακής συμφόρησης αλλά και για λόγους αποφυγής σύγκρουσης, τα πλοία οφείλουν να εισέρχονται και να εξέρχονται από λιμάνια ή ναύσταθμους με μικρή ταχύτητα. Για τον εντοπισμό, λοιπόν, των σκαφών αυτών θα ανατρέξουμε στη συλλογή «nari_dynamic», η οποία περιέχει πληροφορίες σχετικά με την ταχύτητα του σκάφους, το αναγνωριστικό του καθώς και τις γεωγραφικές συντεταγμένες του.

Εύρεση μέσης ταχύτητας σκαφών.

- *show dbs*

- *use aisdata*
- *db.nari_dynamic.aggregate ([{ $group: { _id: null, avgspeed: { $avg: "$speedoverground" }}}])*

```
> db.nari_dynamic.aggregate ( [ { $group: { _id: null, avgspeed: { $avg :"$speedoverground" }}}])
{ "_id" : null, "avgspeed" : 4.15695 }
```

Η μέση ταχύτητα των σκαφών υπολογίστηκε στους 4.15695 κόμβους.

Εύρεση των 2 σκαφών με τη μεγαλύτερη ταχύτητα.

- *db.nari_dynamic.find().sort ({"speedoverground" : -1}).limit(2). pretty()*

Τα δύο σκάφη που επιστρέφει η αναζήτηση είναι αυτά με sourcemmsi: 228190592 με ταχύτητα 51,3 κόμβους και sourcemmsi: 228374000 με ταχύτητα 15.3 κόμβους.

Κάπου εδώ, για να εντοπιστούν τα περαιτέρω στοιχεία των δύο αυτών σκαφών θα πρέπει να ενώσουμε τις πληροφορίες που περιέχονται στη συλλογή «nari_static» (όνομα, τύπος, προορισμός σκάφους) με αυτές που εντοπίσαμε στο προηγούμενο ερώτημα (αναγνωριστικό, ταχύτητα, θέση σκάφους). Η εντολή που χρησιμοποιήθηκε ήταν η *"$ lookup",* η οποία παίζει ρόλο παρόμοιο με αυτόν του join μεταξύ των πινάκων στις σχεσιακές βάσεις δεδομένων.

Έυρεση των στοιχείων του σκάφους με τη μεγαλύτερη ταχύτητα

- *db.nari_dynamic.aggregate ([{ $ match: { "": 9803}},*
  { $ lookup: { from: "nari_static", localField: "sourcemmsi", foreignField: "sourcemmsi", as: "plirofories" }},
  { $ project: { "speedoverground": 1, "lon": 1, "lat": 1, "plirofories": { "shipname":1, "shiptype":1, "destination":1}}}])

Το σκάφος που εντοπίστηκε να πλεέι με ταχύτητα 51.3 κόμβους ονομάζεται ARGONAUTE (γαλλική σημαία) και είναι σκάφος ανεφοδιασμού εγκατασάσεων ανοιχτής θάλασσας έχοντας ως προορισμό το λιμάνι του Brest. Αναζητώντας περισσότερες πληροφορίες σχετικά με αυτό το σκάφος βρέθηκε ότι η μέγιστη δυνατή ταχύτητα που μπορεί να αγγίξει είναι 15,5 κόμβοι. Συμεραίνουμε, λοιπόν, ότι πιθανότατα πρόκειται για εσφαλμένη τιμή.

Σχήμα 2. ARGONAUTE

Το αμέσως επόμενο σκάφος με τη μεγαλύτερη ταχύτητα ήταν το HC JETTE-MARIT που έπλεε με ταχύτητα 15,3 κόμβων. Το HC JETTE MARIT είναι ένα φορτηγό πλοίο που έχει ναυπηγηθεί το 2009 με σημαία Antigua Barbuda. Η χωρητικοτητα του πλοίου είναι 22108 τόνοι και το τρέχον βύθισμά του αναφέρεται ότι είναι 5.8 μέτρα. Το ολικό μήκος του είναι 159.9 μέτρα και το πλάτος του είναι 24 μέτρα. Ο προορισμός του ήταν το κανάλι του En peche.

Σχήμα 3. HC JETTE-MARIT

4.4 Use case 2: indexing

Μια δεύτερη περίπτωση χρήσης του συνόλου δεδοένων μας, είναι ο εντοπισμός των σκαφών που βρίσκονται σε μια συγκεκριμένη περιοχή (γεωγραφικό μήκος και πλάτος). Το να γίνεται γνωστό ποιά σκάφη βρίσκονταν σε μια περιοχή και πότε, είναι ιδιαιτέρως σημαντικό , ειδικά σε περιπτώσεις ατυχημάτων, παραβιάσεων προστατευμένων ζωνών, ακόμα και θαλάσσιων μολύνσεων (παράνομη ρίψη λημμάτων). Στο σημείο αυτό, αξίζει να τονιστεί η σημασία των ευρετηρίων (indexes) ως προς το χρόνο εκτέλεσης των ερωτημάτων.

Με τη χρήση ευρετηρίων, τα ερωτήματα που θέτουμε στην MongoDB μπορούν να εκτελεστούν πολύ αποδοτικά. Όταν δεν υπάρχει κάποιο ευρετήριο που μπορεί να χρησιμοποιηθεί για το ερώτημα μας, τότε η MongoDB είναι αναγκασμένη να διατρέξει όλα τα έγγραφα της συλλογής, ώστε να επιστρέψει εκείνα τα έγγραφα που ικανοποιούν το περιορισμό που θέσαμε. Αντιθέτως, εάν υπάρχει κατάλληλο ευρετήριο που μπορεί να χρησιμοποιηθεί για ερώτημα μας, τότε μειώνεται δραματικά ο αριθμός των εγγράφων που θα επισκεφτεί η MongoDB ώστε να μας απαντήσει.

Ακολουθεί, λοιπόν, ο εντοπισμός των σκαφών που βρίσκονται σε συγκεκριμένα γεωγραφικά μήκη (-4.497^{o} έως -4.496^{o}) και πλάτη (48.382^{o} έως 48.383^{o}). Θα αναδειχθεί η σημασία των indexes, παρατηρώντας τον χρόνο εκτέλεσης του ερωτήματος ΧΩΡΙΣ και ΜΕ ευρετήριο.

Στην περίπτωση μας, επειδή στόχος ήταν να εισάγουμε στο ευρετήριο δύο πεδία (lon, lat) κάναμε χρήση του σύνθετου ευρετηρίου (compound index).

Χωρίς ευρετήριο

- *db.nari_dynamic.find({ $and: [{"lon": {$gte:-4.497, $lte:-4.496}}, {"lat":{$gt:48.382, $lte:48.383}}]}).explain("executionStats")*

Με ευρετήριο

- *db.nari_dynamic.createIndex({"lon":1, "lat":1})*
- *db.nari_dynamic.find({ $and: [{"lon": {$gte:-4.497, $lte:-4.496}}, {"lat":{$gt:48.382, $lte:48.383}}]}).explain("executionStats")*

Παρατηρείται, λοιπόν, ότι στην περιοχή με γεωγραφικές συντεταγμένες που εισάγαμε στο ερώτημα εντοπίζονται 809 σκάφη. Σε περίπτωση που επιθυμούμε να εντοπίσουμε τα αναλυτικά στοιχεία των σκαφών αυτών αρκεί να επαναλάβουμε το ερώτημα χωρίς το επίθεμα «.explain("executionStats")», όπως φαίνεται παρακάτω:

- *db.nari_dynamic.find({ $and: [{"lon": {$gte:-4.497, $lte:-4.496}}, {"lat":{$gt:48.382, $lte:48.383}}]})*

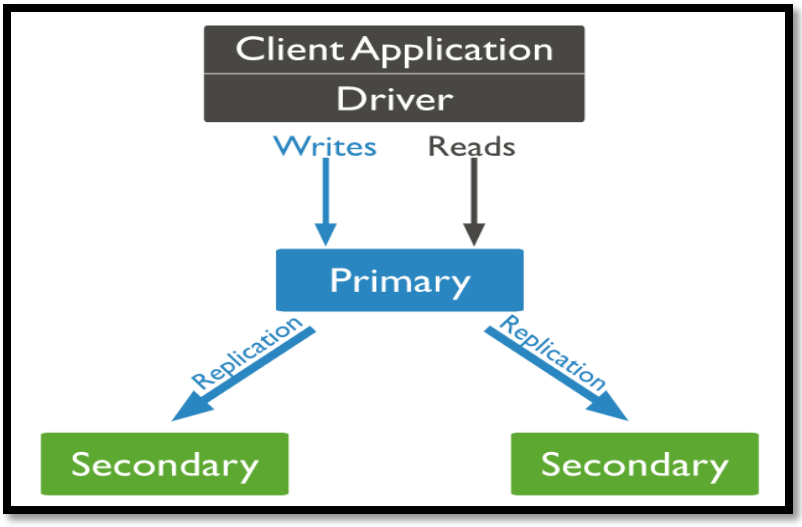
Επίσης, φαίνεται ο χρόνος εκτέλεσης του ερωτήματος να μειώνεται δραστικά με τη χρήση του ευρετηρίου. Πιο συγκεκριμένα, το ερώτημα εκτελέστηκε σε 15 msec χωρίς τη χρήση index, ένω με χρήση αυτού, ο αντίστοιχος χρόνος περιορίστηκε στα 5 msec.

Κεφάλαιο 5ο : Replication-Sharding

5.1 Replication

Βασικός σκοπός του replication είναι η δυνατότητα αποθήκευσης των δεδομένων σε παραπάνω από έναν κόμβους, προσφέροντας έτσι υψηλή διαθεσιμότητα. Πρακτικά αυτό σημαίνει ότι τα δεδομένα εκτός από τη βάση δεδομένων ενός server (primary), αντιγράφονται και στις βάσεις δεδομένων περισσοτέρων server (secondaries), έτσι ώστε σε περίπτωση αποτυχίας του primary server τα δεδομένα να μην χαθούν. Επίσης, κατά αυτόν τον τρόπο, μπορούν να έχουν πρόσβαση στα δεδομένα πολλαπλοί χρήστες.

Η διαδικασία του replication έχει ως εξής. Οι εγγραφές και οι αναγνώσεις των δεδομένων (reads and writes) εκτελούνται από τον κύριο κόμβο, ενώ οι οι δευτερεύοντες κόμβοι αναλαμβάνουν την αντιγραφή των δεδομένων. Το γεγονός αυτό, μειώνει θεαματικά τον χρόνο εκτέλεσης των διαδικασιών, αφού ο primary μπορεί να συνεχίσει να δέχεται εγγραφές, ακόμα και αν δεν έχει ολοκληρωθεί η διαδικασία της αντιγραφής, την οποία και αναλαμβάνουν οι secondaries. Φυσικά, αν για οποιοδήποτε λόγο παρουσιάσει σφάλμα ή πτώση ο κύριος κόμβος (master), αυτόματα, ένας από τους δευτερογενείς (slave), αναλαμβάνει να υποδηθεί το ρόλο του κύριου.



Σχήμα 4. Γραφική αναπαράσταση replication στη MongoDB

5.2 Εφαρμογή replication

Για την υλοποίηση-ανάδειξη της διαδικασίας του replication δημιουργήσαμε τρείς εικονικούς servers στον ίδιο υπολογιστή. Κάθε ένα εικονικός server θα «τρέχει» από διαφορετικό port και θα γίνει προσπάθεια διασύνδεσης των τριών αυτών server, καθιστώντας τον έναν εξ αυτών primary και τους άλλους δύο secondaries. Η παραπάνω διαδικασία θα προσοιμοιάζει, λοιπόν, με τη διαδικασία του replication μεταξύ δύο (ή περισσοτέρων) πραγματικών server.

- Η βάση δεδομένων μας «τρέχει» από τον primary server με port 27017 .
- Δημιουργούμε δύο φακέλους-αντίγραφα με όνομα data1 και data2 με port 27020 και 27021 αντίστοιχα, οι οποίοι θα παίξουν τον ρόλο των secondaries.

  >dbpath =C:\data1\db\path
  logpath=C:\data1\log\mongod.log
  port=27020

  >dbpath =C:\data2\db\path
  logpath=C:\data2\log\mongod.log
  port=27021

- Από το services.mcs παγώνουμε τη λειτουργία του MongoDB server, ο οποίος «τρέχει» από το port 27017
- Με τις ακόλουθες εντολές δημιουργούμε τους 3εικονικούς MongoDB servers, που θα περιέχουν τα ίδια δεδομένα.

  > mongod --dbpath "C:\Program Files\MongoDB\Server\4.0\data" --logpath "C:\Program Files\MongoDB\Server\4.0\log\mongod.log" --port 27017 --storageEngine=wiredTiger --journal --replSet r2schools

  > mongod --dbpath "C:\data1\db" --logpath "C:\data1\log\mongod.log" --port 27020 --storageEngine=wiredTiger --journal --replSet r2schools

> mongod --dbpath "C:\data2\db" --logpath "C:\data2\log\mongod.log" --port 27021 --storageEngine=wiredTiger --journal --replSet r2schools

- Αυτό που μένει να κάνουμε είναι να διασυνδέσουμε τους 3 κόμβους μεταξύ τους με τις ακόλουθες εντολές.

On Primary Server.
> rsconf={_id:r2schools,members:[{_id:0,host:"localhost:27017"}]}
> rs.initiate(rsconf)
> rs.add("localhost:27020")
> rs.add("localhost:27021")

- Έτσι, ο κόμβος localhost: 27017 είναι ο primary και οι άλλου δύο οι secondaries.
- Η διαδικασία του replication έχει ολοκληρωθεί αφού οι βάσεις δεδομένων μας εμφανίζονται πλέον και από τους 3 κόμβους.
- Για επαλήθευση, συνδεόμαστε στον κόμβο 27020 και με την εντολή *rs.slaveOk()*, αναζητούμε στον secondary server πια, μια τυχαία εγγραφή. Όντως η εγγαφή αυτή εμφανίζεται στην οθόνη μας.

5.3 Sharding

Με τον όρο sharding αναφερόμαστε σε μία μέθοδο διαμοιρασμού των δεδομένων σε πολλές μηχανές. Η MongoDB χρησιμοποιεί το sharding, για να υποστηρίξει πολύ μεγάλο όγκο δεδομένων αλλά να έχει υψηλότερη απόδοση, κι αυτό διότι ορισμένα συστήματα βάσεων δεδομένων με μεγάλο όγκο δεδομένων ή εφαρμογές με υψηλό throughput μπορεί να επηρεάζουν αρνητικά την επίδοση ενός διακομηστή.

Υπάρχουν δύο διαφορετικοί μέθοδοι για την ανάπτυξη ενός συστήματος, η κατακόρυφη και η οριζόντια κλιμάκωση (scaling). Παρακάτω ακολουθεί μια σύντομη περιγραφή των δύο μεθόδων:

Η κατακόρυφη κλιμάκωση (vertical scaling), επιφέρει την αύξηση της χωρητικότητας ενός μόνο διακομιστή, όπως για παράδειγμα τη χρήση μίας πιο ισχυρής CPU, τη προσθήκη περισσότερης μνήμης RAM ή την αύξηση του χώρου αποθήκευσης. Ωστόσο, αυτό δεν είναι πάντα εφικτό, καθώς εξαρτάται και από το αν υποστηρίζεται από την εκάστοτε τεχνολογία κάθε μηχανήματος. Συνεπώς, υπάρχει ένα πρακτικό μέγιστο για κατακόρυφη κλιμάκωση.

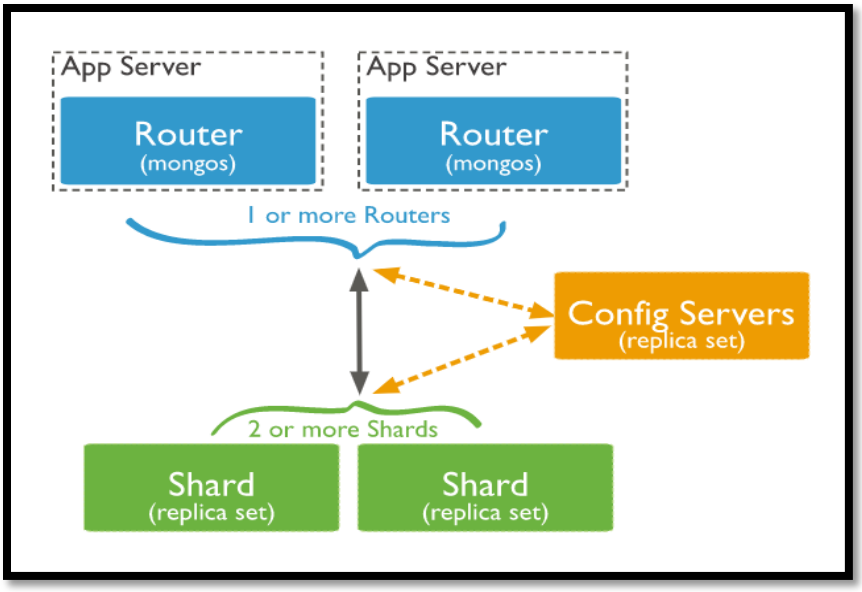
Η οριζόντια κλιμάκωση (horizontal scaling) περιλαμβάνει το διαμοιρασμό ενός συνόλου δεδομένων και του φόρτίου σε πολλαπλούς διακομιστές, προσθέτοντας επιπλέον διακομιστές, ώστε να αυξηθεί η χωρητικότητα του συστήματος, όπου απαιτείται. Έτσι, ενώ η συνολική ταχύτητα ή η χωρητικότητα ενός μηχανήματος μπορεί να μην είναι υψηλή, κάθε μηχάνημα χειρίζεται ένα υποσύνολο του συνολικού φόρτου εργασίας, παρέχοντας, πιθανότατα, καλύτερη απόδοση από ένα μόνο διακομιστή υψηλής ταχύτητας και υψηλής χωρητικότητας. Για να είναι αυτό εφικτό, απαιτείται μόνο η ανάπτυξη επιπλέον διακομιστών σε σχέση με τις ανάγκες του συστήματος. Τελικά, το αποτέλεσμα μπορεί να είναι πιο οικονομικό σε σχέση με τη χρήση υλικών υψηλής τεχνολογίας για ένα μόνο μηχάνημα. Ωστόσο, υπάρχει αυξημένη πολυπλοκότητα τόσο στην υποδομή όσο και στην συντήρηση του συστήματος.

Η MongoDB υποστηρίζει οριζόντια κλιμάκωση μέσω του sharding. Ένα sharded cluster της MongoDB αποτελείται από τα ακόλουθα στοιχεία:

- shard: κάθε shard περιέχει ένα υποσύνολο δεδομένων και μπορεί να υλοποιηθεί με τη χρήση ενός replication set.

- mongos: κάθε mongos λειτουργεί σαν ένα router και παρέχει μία διεπαφή μεταξύ των client εφαρμογών και των sharded clusters.

- config servers: αποθηκεύει τα metadata και διαμορφώνει τις ρυμίσεις για το cluster.



Σχήμα 5. Αλληλεπίδραση των διάφορων εξαρτημάτων ενός shared cluster.

Παρακάτω παρατίθενται οι εντολές που χρησιμοποιήθηκαν για τη διαδικασία του sharding στο σύνολο δεδομένων μας :

- *mongos> use aisdata*
- *mongos> db.nari_dynamic.getShardDistribution() ## επιστρέφει τα shards εάν υπάρχουν*
- *mongos> sh.enableSharding("aisdata") ## επιτρέπει την εκτέλεση του sharding*
- *mongos> sh.shardCollection("aidata.nari_dynamic", {"lon":1, "lat":1})*

Κεφάλαιο 6ο: Συμπεράσματα

Στην παρούσα εργασία μελετήθηκε η σημασία των δεδομένων που προέρχονται από το σύστημα αυτόματης αναγνώρισης (AIS), καθώς επίσης ο τρόπος αποθήκευσης και επεξεργασίας τους με τη χρήση του συστήματος μη σχεσιακής διαχείρισης βάσεων δεδομένων MongoDB. Εξετάστηκαν δύο βασικές περιπτώσεις χρήσης (usecases 1,2) του συνόλου δεδομένων, ενώ παράλληλα αναδείχθηκαν βασικές τεχνικές και λειτουργίες που διαθέτει το μή σχεσιακό σύστημα διαχείρισης της MongoDB (aggregation, indexing, replication, sharding). Τέλος, για μελλοντική μελέτη, θα παρουσίαζε ενδιαφέρον η επανάληψη των ερωτημάτων του usecase 2 (εύρεση σκαφών σε συγκεκριμένη περιοχή), κάνοντας χρήση των γεωχωρικών ευρετηρίων (geospatial indexes) που διατίθενται (σσ. λόγω περιορισμένου χρόνου δεν συμπεριλήφθηκε στην εργασία).

Βιβλιογραφία

[1] http://www.gdmc.nl/publications/2016/Managing_Historic_AIS_data.pdf

[2] https://zenodo.org/record/1167595

[3] http://artemis.cslab.ece.ntua.gr:8080/jspui/bitstream/123456789/13339/1/DT2016-0322.pdf

[4] https://evdoxos.ds.unipi.gr/modules/document/index.php?course=DSERV138&openDir=/5db5e52azTaw

[5] https://evdoxos.ds.unipi.gr/modules/document/index.php?course=DSERV138&openDir=/5db5e52azTaw/5dc18458BAnx

[6] https://www.marinetraffic.com/el/ais/details/ships/shipid:146331/mmsi:304091000/imo:9509255/vessel:HC%20JETTE%20MARIT

[7] https://bourbon-offshore-greenmar.mu/sites/default/files/documents-associes/pdf/argonaute-132-mt-bp-dp1.pdf

[8] https://docs.mongodb.com/manual/replication/

[9] https://docs.mongodb.com/manual/sharding/