ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Π.Μ.Σ. ΔΙΚΤΥΟΚΕΝΤΡΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Διπλωματική Εργασία

NoSQL databases : Ποιοτική και Ποσοτική σύγκριση μεταξυ των CouchDBκαι MongoDB

ΜΥΣΤΡΙΩΤΗ ΑΙΚΑΤΕΡΙΝΗ (ΜΕ11063)

Επιβλέπων Καθηγητής: Μ. Θεμιστοκλέους

Ακαδημαικό Έτος 2014-2015

ΠΕΡΙΛΗΨΗ – ABSTRACT

Κατά την διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίαςπραγματοποιήσαμε μια εκτενή μελέτη και ανάλυση πάνω σε NoSQL βάσεις δεδεμένων και συγκεκριμένα DocumentOriented βάσεις δεδομένων.

Οι 2 προς μελέτη και έρευνα βάσεις δεδομένων είναι οι CouchDB και η MongoDB.

Μελετώντας λοιπόν την σχετική βιβλιογραφία, μέσα από πηγές στο διαδίκτυο, αντίστοιχες μελέτες και έρευνες συλλέξαμε όλα τα τεχνικά χαρακτηριστικά των βάσεων, τόσο ποιοτικά όσο και ποσοτητά και πραγματοποιήσαμε μια Ποιοτική και Ποσοτική Ανάλυση ανάμεσα σε αυτές τις 2 βάσεις δεδομένων.

Τέλος καταλήξαμε σε κάποια συμπεράσματα σχετικά με την σύγκριση αναμεσα σε αυτές τις βάσεις.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τους καθηγητές μου, τους στενούς φίλους μου και την οικογένεια μου που με στήριξαν κατα την διάρκεια όλων των ετών την φοίτησης μου καθώς και κατά την διάρκεια της εκπόνησης της παρούσας διπλωματικής εργασίας.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΕΦΑΛΑΙΟ 1 – ΕΙΣΑΓΩΓΗ

1.1Σκοπός και αντικείμενο της εργασίας

Ο σκοπός της παρούσας εργασίας είναι να αναδείξει με κάθε λεπτομέρεια την τεχνολογία των υπομελέτη βάσεων δεδομένων και συγκεκριμένα να αναλυθεί:

- Η αρχιτεκτονική
- Η τοπολογία
- Τα ποιοτικά χαρακτηριστικά
- Τα ποσοτικά χαρακτηριστικά

Έχοντας λοιπόν μια συμπαγή θεωρητικη βάση, θα εκπονηθεί μια αναλυτική ποιοτική και ποσοτική ανάλυση, συγκρίνοντας τα χαρακτηριστικά των βάσεων.

ΚΕΦΑΛΑΙΟ 2 – ΑΝΑΣΚΟΠΗΣΗ ΒΙΒΛΙΟΓΡΑΦΙΑΣ

2.1 ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΤΕΧΝΟΛΟΓΙΑ NoSQL ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Ο όρος “Document-OrientedDatabase”, είναι το λογισμικό αυτό το οποίο έχει σχεδιαστεί για αποθήκευση, εξόρυξη και διαχείριση document-oriented πληροφορίας, όρου εξίσου γνωστού και ως “semi-structureddata”. Το εν λόγω λογισμικό, αποτελεί μια από τις βασικές κατηγορίες της ευρύτερης οικογένειας των “document-oriented” βάσεων δεδομένων.

Σε αντίθεση με τις «παραδοσιακές» σχεσιακές βάσεις δεδομένων, στις οποίες βασικές έννοιες είναι οι πίνακες και οι συσχετίσεις, τα no-sql σχήματα, έχουν σχεδιαστεί με βάση την πιο αφηρημένη έννοια του «document».

Η κεντρική ιδέα γύρω από τις document-orienteddatabases είναι ότι περιέχουν μεγάλο όγκο δεδομένων ο οποίος μπορούν να χρησιμοποιηθεί. Οι documentoriented υλοποιήσεις βάσεων δεδομένων, διαφέρουν κατα πολύ από τις παραδοσιακές υλοποιήσεις σχεσιακών βάσεων, αναφορικά με την λειτουργικότητα. Οι περισσότερες DO βάσεις δεδομένων, υποστηρίζουν documents σε διαφορετικά format και μορφή, και τα ενσωματώνουν σε ένα εσωτερικά τυποποιημένο format, ενώ ταυτόχρονα αποθηκεύουν κάποια σημαντικά dataitems, τα οποία είναι συνδεδεμένα με το document.

Ενα χαρακτηριστικό παράδειγμα θα μπορούσε να είναι το εξής: Σάρωση εγγράφων, εξαγωγή τίτλου, συγγραφέα, και ημερομηνία, είτε χρησιμοποιώντας κάποιο OCR λογισμικό, είτε έχοντας dataentry, και εποθηκεύοντας κάθε document σε μια σχεσιακή βάση 4 στηλών (τίτλος, συγγραφέας, ημερομηνία και ένα blob πεδίο με τις φωτογραφίες των σελιδων. Μερικές documentorienteddatabases, κάνουν το ίδιο ακριβώς πράγμα αλλά αντί για φωτογραφίες κειμένου αποθηκεύουν έγγραφα pdf.

Ένα μεγάλο ποσοστό της εξόρυξης των δεδομένων γίνεται μέσω HTML, XML, και TeX, είτε μέσω συστημάτων τα οποία τουλάχιστον υποστηρίζουν μηχανισμούς export στις εκ των άνωθι μορφές. Σε πραγματικές συνθήκες, άλλες μορφές documents είναι email, των οποίων το format υποστηρίζει την χρήση metadata στο header. Σε τέτοιες περιπτώσεις, το documentoriented λογισμικό δεν έχει μόνο πρόσβαση σε φωτογραφίες, αλλά και στο κείμενο, και πιο συγκεκριμένα σε συγκεκριμένες λέξεις σε διάφορα sections του body του email ("footnote", "chapter", "authorname", κ.λπ.). Όλη αυτή η πληροφορία είναι διαθέσιμη για αναζήτηση, στατιστική ανάλυση και reporting καθώς και εξόρυξη δεδομένων. Ακόμα και όταν τα δεδομένα δεν βρίσκονται σε μια τόσο ξεκάθαρη δομή, υπάρχει πάντα η δυνατότητα χρήσης διάφορων άλλων διαθέσιμων components μέσω ευρετικών και άλλων μεθόδων.

Τα documents από την άλλη πλευρά, είναι δομημένα με τέτοιο τρόπο έτσι ώστε να είναι προσβάσιμα και διαχειρίσιμα τόσο από τους χρήστες όσο και από υπολογιστές. Η δομή τους χαρακτηρίζεται τόσο από μια εξαιρετικά βολική επαναχρησιμοποίηση μικρών components (λέξεις και φράσεις, αλλά και παραγράφους και υποσημειώσεις), και από την ελεύθερη μίξη των τύπων αυτών, τεχνική η οποία δεν χρησιμοποιείται σε παραδοσιακά σχεσιακά σχήματα. Γιαπαράδειγμα: ανυποθέσουμεόχιΧείναιέναέγγραφο, τοοποίοαποτελείταιαπόδομικέςμονάδες Hamlet is a document, consisting of structural units such as acts, scenes, speeches, attributions, stage directions, and notes. An entry in one's smart-phone address book is a "document" but only barely so, resembling a single record in a relational or similar database far more.

Γιαταεξαγόμενα metadata, οποιοδήποτε format είναιαποδεκτό, όπως XML, YAML, JSON, and BSON. Ωστόσο, το έγγραφο αυτό καθ'εαυτό, αποθηκεύεται συνήθως στην βάση, τουλάχιστον σε ένα blob πεδίο στην αρχική του μορφή (XML, PDF, binaryformats, είτε plaintext), λειτουργικότητα η οποία είναι άμεσα συνδεδεμένη με το format των δεδομένων και την δυνατότητα της βάσης να εξάγει δεδομένα από αυτό το format.

Τα documents μιας documentoriented βάσεως είναι παρόμοια σε πολλές περιπτώσεις, με εγγραφές σε παραδοσιασκά σχεσιακά σχήματα αλλά έχουν μια περισσότερο «εσωτερική» δομή, υπό την ένοια ότι η κάθε documentoriented βάση, μπορεί να «γνωρίζει» τον όγκο των δεδομένων, και να τα χρησιμοποιήσει αναλόγως. Τα documents, ιδιαιτέρως σε XML, TeX μορφή, τηρούν τις αρχές ενός σχήματος, αλλά άλλες μορφές όχι, και σε περίπτωση που τηρούν, το σχήμα αυτό δεν είναι σαφές.

Ως παράδειγμα μπορούμε να δούμε το παρακάτω document, σε XML μορφή.

```
<Article>
<Author>
<FirstName>Bob</FirstName>
<Surname>Smith</Surname>
</Author>
<Abstract>This paper concerns....</Abstract>
<Section n="1"><Title>Introduction</Title>
<Para>...
</Section>
</Article>
```

Ένα 2ο document, της ίδιας δομής και σχήματος, μπορεί να έχει διαφορετικό αριθμό και δόμηση στα επιμέρους τμήματα του, καθώς και επαναλαμβανόμενη καθώς και επιπρόσθετη πληροφορία που το 1ο έγγραφο δεν διαθέτει. Σε αντίθεση με μια σχεσιακή δομή, όπου κάθε εγγραφή περιέχει πανομοιότυπη αλληλουχία πεδίων (μερικά από τα οποία μπορεί να είναι κενά, να μην έχεο)

Τα 2 αυτά documents τυπικώς διαθέτουν πολλά κοινά στοιχεία, αλλά το καθένα ξεχωριστά μπορεί να περιέχει elements που άλλα μπορεί να μην διαθέτουν. Σε αντίθεση με τις σχεσιακές βάσεις δεδομένων, όπου κάθε εγγραφή περιέχει μια πανομοιότυπη ακολουθία πεδίων, οι δομές στις documentoriented βάσεις γενικώς επιτρέπουν έναν απεριόριστο αριθμό ιεραρχικώς οργανωμένων components, με εκτενώς επαναλαμβανομένων.

Θα ήταν παράλογο για παράδειγμα να σχεδιαστεί μια βάση δεδομένων με έναν πίνακα «Sections» ο οποίος θα περιείχε τόσα πεδία όσο και ο αριθμός των παραγράφων. Ακόμα και αν κάτι τέτοιο ήταν εφικτό, η ονομασία των σχετιζόμενων πεδίων σαν ”p1”, “p2”, “p3” κ.ο.κ δεν υποδεικνύει σε καμία περίπτωση ότι αυτά τα πεδία έχουν κάποια σχέση μεταξύ τους είτε έχουν κάποια άλλη σημασιολογική συνοχή. Προκειμένου λοιπόν να αποφευχθεί η σύγχιση σχετικά με τον όρο «πεδίο» μια βάσεως, στις documentoriented βάσεις τα πεδία αναφέρονται ώς “components” ή “elements”.

Κλειδιά και Εξόρυξη

Τα documents μπορεί να εισαχθούν στην βάση μέσω ενός μοναδικού κλειδιού το οποίο αναπαριστά το document. Αυτό το κλειδί μπορεί να είναι είτε ένα string, είτε ένα URI, είτε ένα path. Τυπικώς η βαση διατηρεί ένα index του κλειδιού έτσι ώστε να επιταχύνει την ανάκτηση του document. Οι πρώτες μορφές documentoriented βάσεων δεδομένων κάνουν κάτι περισσότερο από indexing όμως, οι νεότερες παρέχουν περισσότερα features καθώς εξάγουν και κάνουν indexing σε όλα τα metadata του document καθώς και σε όλο το περιέχόμενο του document. Τέτοιου είδους βάσεις, διαθέτουν μια γλώσσα ερωτημάτων η οποία επιστρέπει στους χρήστες να ανακτήσουν documents βάσει του περιεχομένου τους. Για παράδειγμα, αν κάποιος χρήστης θέλει να ανακτήσει όλα τα documents των οποίων η ημερομηνία είναι μεταξύ ενός range, που περιέχουν παραπομπή σε ένα άλλο document κ.λπ.. Το σύνολο των APIS είτε των γλωσσών ερωτημάτων τα οποία είναι διαθέσιμα, όπως και η αναμενόμενη απόδοση των ερωτημάτων ποικίλει από υλοποίηση σε υλοποίηση.

Οι διάφορες υλοποιήσεις προσφέρουν μια ποικιλία τρόπων προκειμένουν να οργανωθούν τα δεδομένα συμπεριλαμβανομένων των παρακάτω:

- Collections
- Tags
- Non-visible Metadata

- Directory hierarchies
- Buckets

2.2. MongoDB

Η δύναμη της MongoDB έγκειται στην ευελιξία, την δύναμη, την ευκολία χρήσης, και την ικανότητα να χειριστεί μεγάλες και μικρές εργασίας.

Κυκλοφόρησε για πρώτη φορά στο κοινό το 2009, και αποτελεί μια ταχύτατα ανερχόμενη τεχνολογία στον κόσμο της NoSQL. Ο αρχικός της σχεδιασμός την ώθησε να χαρακτηριστεί ως μια επεκτάσιμη βάση δεδομένων - το όνομα Mongo προέρχεται από το "γιγαντιαία" -με απόδοση και εύκολη πρόσβαση σε δεδομένα και τους στόχους του πυρήνα του σχεδιασμού. Πρόκειται για μια βάση δεδομένων εγγράφων, η οποία επιτρέπει στα δεδομένα να εμμένουν σε μια ένθετη κατάσταση, και το σημαντικότερο, επιτρέπει την εκτέλεση queries πάνω σε αυτά τα ένθετα δεδομένα με έναν ad-hoc τρόπο. Επιβάλλει no schema πολιτική (παρόμοια με το Riak αλλά σε αντίθεση με Postgre), έτσι ώστε τα έγγραφα να μπορούν προαιρετικά να περιέχουν πεδία ή τα είδη που περιέχει να μην περιέχεται σε κανένα άλλο έγγραφο στη συλλογή.Σε καμία περίπτωση όμως η ευελιξία της Mongo δεν την κάνει εύκολη. Στην αγορά υπάρχουν πολλές υλοποιήσεις της Mongo όπως το Foursquare, bit.ly και το CERN, για τη συλλογή μεγάλου όγκου δεδομένων Hadron Collider.

Τα κύρια χαρακτηριστικά της Mongo κινούνται ανάμεσα στο ισχυρό queryability μιας σχεσιακής βάσης δεδομένων και την κατανεμημένη φύση των άλλων datastores όπως το Riak ή του HBase.Η Mongo είναι μια JSON βάση δεδομένων εγγράφου (αν και τα τεχνικά δεδομένα που είναι αποθηκευμένα σε δυαδική μορφή JSON είναι γνωστή ως BSON). Ένα έγγραφο Mongo μπορεί να παρομοιαστεί με μια σχεσιακή σειρά του πίνακα χωρίς σχήμα, του οποίου οι τιμές μπορούν να αποθηκεύονται σε ένα αυθαίρετο βάθος.Η Mongo είναι μια εξαιρετική επιλογή για μια συνεχώς αυξανόμενη κατηγορία έργων στο διαδίκτυο με τις απαιτήσεις αποθήκευσης δεδομένων μεγάλης κλίμακας, αλλά πολύ μικρό προϋπολογισμό για να αγοράσει κάποιος το hardware. Χάρη στην έλλειψη δομημένων σχημάτων, στην Mongo μπορούν να αναπτυχθούν εφαρμογές και να αλλάξουν μαζί με το μοντέλο των δεδομένων. Ειδικότερα όταν πρόκειται για νεοσύστατους οργανισμούς, όπου υπάρχει η τάση επέκτασης τοτε είναι επιτακτική η ανάγκη αναβάθμισης του hardware προτίστως, η Mongo είναι η πιο ασφαλής επιλογή.

Η Mongo λόγω της ευελιξίας των ad-hocqueries που παρεχει και της ικανότητας της να είναι προσαρμόσιμη σε web-oriented επεκτάσεις είναι μια ιδανική λύση. Πέραν αυτών όμως, ακόμα ένα δυνατό χαρακτηριστικό της αποτελεί το δυνατό support community. Χωρίς να υποστηρίζει ότι δεν υπάρχουν μειονεκτήματα, υπάρχει ένα σαφές σχέδιο για τις περιπτώσεις χρησης αναλόγως με την υποδομή παρόλη την επεκτατικη διάθεση. Μια trade-off αυτής της εξελικτικής συμπεριφοράς είναι ότι υπάρχει μεγάλη σιγουριά εμπιστοσύνη για επεκταση στήριξη στην τεχνολογία της Mongo.

Δεδομένου ότι η Mongo δεν έχει schema, δεν υπάρχει καμία ανάγκη να καθοριστεί κάτι από πριν:

Σε αντίθεση με μια σχεσιακή βάση δεδομένων, η Mongo δεν υποστηρίζει server-side joins. Με ένα μόνο JavaScript script θα ανακτήσει ένα έγγραφο και όλα τα ένθετα του περιεχομένου του.

"_id" : ObjectId("4d0ad975bb30773266f39fe3"), → το autonumbering του συστήματος είναι ότι κάθε διεργασία σε κάθε μηχάνημα μπορεί να χειριστεί το δικό του ID generation χωρίς να υπάρξει conflict με άλλες υλοποιήσεις mongodb. Αυτή η σχεδιαστική επιλογή δίνει μια ιδέα της κατανεμημένης φύσης του Mongo. Η βασικη γλώσσα στην Mongo είναι η JavaScript. Όπως και στην PostgreSQL, έτσι και στην Mongo υπάρχει η δυνατοτητα δημιουργίας ad-hoc ερωτημάτων, με τιμές πεδίων ή περιοχών ή έναν συνδυασμό κριτηρίων.Η πραγματική δύναμη της Mongo όμως προέρχεται από την ικανότητα της τεχνολογιας να ψάξει κάτι από ένα έγγραφο και να επιθστρέψει αποτελέσματα σε δευτερεύοντα.

Η Mongo δεν λειτουργεί με βάση τα χαρακτηριστικά, έχει μόνο μια εσωτερική, σιωπηρή κατανόηση των χαρακτηριστικών για λόγους βελτιστοποίησης. Αλλάτίποταγιατο interface δενείναι attribute- oriented. Η Mongo είναι document-oriented.

Η Mongo επίσης δεν είναι κατασκευασμένη για να εξυπηρετεί joins. Λόγω της κατανεμημένης φύσης της, τα joins είναι αρκετά αναποτελεσματικές πράξεις. Ακόμα, είναι μερικές φορές χρήσιμο για τα έγγραφα ν' αναφέρονται το ένα στο άλλο.

Ένα έγγραφο μπορεί να οραματιστεί ως ένα σχήμα στο σχεσιακό μοντέλο, και κλειδώνεται από ένα παραγόμενο _id. Μια σειρά από έγγραφα που ονομάζεται collection στο Mongo, παρόμοια με έναν πίνακα στην PostgreSQL.

Σε αντίθεση με τις προηγούμενες μορφές που έχουμε αντιμετωπίσει, με τα collection των συνόλων των απλών τύπων δεδομένων, η Mongo αποθηκεύει complex, denormalized έγγραφα. Η Mongo αποτελεί επιστέγασμα αυτής της ευέλικτης αποθήκευσης με έμμεση στρατηγική με ένα ισχυρό μηχανισμό query δεν περιορίζεται από οποιοδήποτε προκαθορισμένο σχήμα.

Είναι η denormalized φύση της που την καθιστά ένα document datastore, μια θαυμάσια επιλογή για την αποθήκευση δεδομένων με άγνωστες ιδιότητες, ενώ άλλες μορφές (όπως σχεσιακές ή σε στήλες) προαπαιτούν την εκ των προτέρων γνώση του σχήματος και απαιτούν schema migrations για την προσθήκη ή να επεξεργασία πεδίων.

Ένα από τα χρήσιμα ενσωματωμένα χαρακτηριστικά της Mongo είναι το indexing για να αυξηθεί η απόδοση των queries, κάτι που, όπως έχουμε δει, δεν είναι διαθέσιμο για όλες τις βάσεις δεδομένων NoSQL. Η MongoDB παρέχει αρκετές από τις καλύτερες δομές δεδομένων για indexing, όπως το κλασικό B-tree, και άλλες προσθήκες, όπως δισδιάστατα και σφαιρικά Geospatial ευρετήρια.

Κάθε φορά που δημιουργείται μια νέα συλλογή, η Mongo δημιουργεί αυτόματα ένα ευρετήριο από το _id. Οι δείκτες αυτοί μπορούν να βρεθούν στη συλλογή system.indexes. Τα δεδομένα είναι ήδη χαρτογραφημένα σε μια υπάρχουσα συλλογή των εγγράφων. Η χαρτογράφηση δεν είναι απαραίτητη, απλά να μειώνει τα έγγραφα.

Η λειτουργία του group() είναι ισχυρή, όπως το GROUP στην SQL BY-αλλά η εκτέλεση της function στην Mongo έχει ένα μειονέκτημα. Κατ 'αρχάς, θα περιορίζονται σε ένα αποτέλεσμα 10.000 εγγράφων. Επιπλέον, ο κατακερματισμός μιας συλλογής στην Mongo (το group()) δεν θα λειτουργήσει.

Γραμμένη σε C++, τα χαρακτηριστικά της MongoDB είναι:

⇢ Document-Oriented Storage » JSON-style documents με δυναμική σχηματική προσφορά απλότητας και δύναμης.

⇢ Full Index Support » Δείκτης για κάθε χαρακτηριστικό.

⇢ Replication & High Availability » Mirror σεόλατα LAN’s και WAN’s.

⇢ Auto-Sharding » οριζόντια κλίμακα, χωρίς να θίγονται οι λειτουργίες.

⇢ Querying » Πλούσιακαιευάλικτα, document-based queries.

⇢ Fast In-Place Updates » Ατομικοίτροποποιητέγια contention-free απόδοση.

⇢ Map/Reduce » Ευέλικτη συσσώρευση και επεξεργασία δεδομένων.

⇢ GridFS » Αποθήκευση αρχείων σε οποιοδήποτε μέγεθος χωρίς να περιπλέκονται τα stack.

⇢ MongoDB Management Service » Διαχείριση της MongoDB σε cloud υποδομή της επιλογή σας.

⇢ MongoDB Enterprise » Ο καλύτερος τρόπος να τρέξετε την MongoDB σε παραγωγή. Ασφαλές, με Υποστήριξη και Πιστοποιημένα.

⇢ Production Support » Δίνοντας πρόσβαση στο παγκόσμιο μοντέλο υποστήριξης 24x365.

Το MongoDB αποθηκεύει δεδομένα σε μορφή εγγράφων, τα οποία είναι JSON-like field and value pairs. Έγγραφα τα οποία αναλογικά σε υποδομή σε γλώσσες προγραμματισμού οι οποίες συνδέουν τα κλειδιά με τιμές (π.χ λεξικά, hashes, χάρτες, και συνειρμικές συστοιχίες).

Επισήμως, τα έγγραφα MongoDB είναι έγγραφα BSON. Το BSON είναι μια δυαδική αναπαράσταση του JSON με πρόσθετους τύπους πληροφοριών. Στα έγγραφα, η τιμή ενός πεδίου μπορεί να είναι οποιοδήποτε από τους τύπους δεδομένων BSON, συμπεριλαμβανομένων άλλων εγγράφων, συστοιχίες και συστοιχίες των εγγράφων.

Το MongoDB αποθηκέυει όλα τα΄έγγραφα σε συλλογές. Μια συλλογή είναι μια ομάδα σχετικών εγγράφων όπου έχουν μια σειρά από κοινά index. Οι συλλογές αυτές είναι ανάλογες με έναν πίνακα σε σχεσιακές βάσεις δεδομένων.

Έγγραφα

Η MongoDB αποθηκέυει ολα τα δεδομένα σ'έγγραφα, τα οποία είναι JSON-style κατασκευής δεδομένων αποτελούμενες από field-and-value pairs:

```
{ "item": "pencil", "qty": 500, "type": "no.2" }
```

Οι περισσότερες δομές δεδομένων προσπελάσιμες από το χρήστη στο MongoDB είναι τα έγγραφα, μεταξύ των οποίων:

⇢ Όλες οι εγγραφές της βάσης δεδομένων.

⇢ Query selectors, τα οποία καθορίζουν τι αρχεία να επιλέξετε για ανάγνωση, ενημέρωση και λειτουργίες διαγραφής.

⇢ Update definitions, τα οποία καθορίζουν τι πεδία τροποποιούνται στην διάρκεια μιας ενημέρωσης.

⇢ Index specifications, οι οποίες καθορίζουν ποιους τομείς να κατατάξουν σε πίνακα.

⇢ Έξοδος δεδομένων από την MongoDB για την υποβολή εκθέσεων και την διαμόρφωση, όπως η έξοδος Serverstatus και το αντίγραφο του εγγράφου διάταξης.

Τύπος Εγγράφου

Το MongoDB αποθηκέυει έγγραφα στον δίσκο σε μορφή BSON serialization. Το BSON είναι μια δυαδική εκπροσώπηση των εγγράφων JSON, αν και περιέχει περισσότερους τύπους δεδομένων από το JSON. για το BSON spec, δείτε bsonspec.org. Δείτε επίσης τύποι BSON.

Το Mongo JavaScript cell και οι οδηγοί γλώσσας του MongoDB μεταφράζουν μεταξύ BSON και ειδικών για τη γλώσσα παράστασης εγγράφων.

Υποδομή Εγγράφων

Τα MongoDB έγγραφα αποτελούνται από field-and-value ζευγάρια και έχουν την ακόλουθη δομή:

MongoDB documents are composed of field-and-value pairs and have the following structure:

```
{
   field1: value1,
   field2: value2,
   field3: value3,
   ...
   fieldN: valueN
}
```

Η τιμή ενός πεδίου μπορεί να είναι οποιοδήποτε από τους τύπους δεδομένων BSON, συμπεριλαμβανομένων άλλων εγγράφων, συστοιχίες, και συστοιχίες των εγγράφων. Το ακόλουθο έγγραφο περιέχει τιμές διαφόρων τύπων:

Οι παραπάνω τομείς έχουν τους ακόλουθους τύπους δεδομένων:

⇢ _id αναφέρεται σε ένα ObjectId.

⇢ name το όνομα ενός embedded εγγράφου το οποίο περιέχει τους τομείς πρώτου και τελευταίου.

⇢ birth and death έχει values τύπου Date.

⇢ Συνεισφορές όπου κατέχουν μια σειρά από strings.

⇢ views έχειτιμή NumberLong τύπου.

Field Names

Τα ονόματα των πεδίων είναι strings.

Έγγραφα που έχουν τους ακόλουθους περιορισμούς σχετικά με τα ονόματα των πεδίων:

⇢ Το πεδίο name _id προορίζεται για χρήση ως πρωτεύον κλειδί, η αξία του πρέπει να είναι μοναδική μέσα στη συλλογή. είναι αμετάβλητη, και μπορεί να είναι οποιουδήποτε τύπου, εκτός από sting.

⇢ Τα ονόματα των πεδίων δεν μπορούν να ξεκινήσουν με χαρακτήρα σύμβολο του δολαρίου ($).

⇢ Τα ονόματα των πεδίων δεν μπορούν να περιέχουν τον χαρακτήρα τελεία (.).

⇢ Τα ονόματα των πεδίων δεν μπορούν να περιέχουν τον null χαρακτήρα.

Τα έγγραφα BSON μπορούν να έχουν περισσότερα από ένα πεδίο με το ίδιο όνομα. Τα περισσότερα MongoDB interfaces, ωστόσο, αντιπροσωπεύουν την MongoDB με δομή (π.χ. πίνακα hash) που δεν υποστηρίζει διπλά ονόματα ενός τομέα. Εάν πρέπει να διαχειρισθούν τα έγγραφα που έχουν περισσότερα από ένα πεδίο με το ίδιο όνομα, τότε μια αναφορά στο documentation του προγράμματος οδήγησης για τον οδηγό της βάσεως.

Ορισμένα έγγραφα που δημιουργούνται από τις εσωτερικές διαδικασίες της MongoDB μπορεί να έχουν διπλά πεδία, αλλά καμία MongoDB διαδικασία δεν θα προσθέσει ποτέ διπλά πεδία σε ένα υπάρχον έγγραφο του χρήστη.

Document Limitations

Τα έγγραφα έχουν τα εξής χαρακτηριστικά:

Όριο Μεγέθους του εγγράφου.

Το μέγιστο μέγεθος ενός εγγράφου BSON είναι 16 megabytes.

Το μέγιστο μέγεθος ενός εγγράφου, βοηθά στο να διασφαλιστεί ότι ένα ενιαίο έγγραφο δεν μπορεί να χρησιμοποιήσει υπερβολική RAM ή, κατά τη διάρκεια της μετάδοσης, ή υπερβολική ποσότητα του εύρους ζώνης. Για να αποθηκευτούν έγγραφα μεγαλύτερα από το μέγιστο μέγεθος, η MongoDB παρέχει το GridFS API.

Πεδίο Τάξης Εγγράφου

Το MongoDB διατηρεί τη σειρά των πεδίων των εγγράφων σύμφωνα με τις λειτουργίες εγγραφής, εκτός από τις ακόλουθες περιπτώσεις:

Το πεδίο _id είναι πάντα το πρώτο πεδίο στο έγγραφο.

Updates που περιλαμβάνουν μετονομασία των ονομάτων του τομέα μπορούν να οδηγήσουν στην αναδιάταξη των πεδίων στο έγγραφο.

Ξεκινώντας από την έκδοση 2.6, η MongoDB προσπαθεί ενεργά να διατηρήσει την τάξη των πεδίων σε ένα έγγραφο. Πριν από την έκδοση 2.6, η MongoDB δεν προστατεύει ενεργά τη σειρά των πεδίων σε ένα έγγραφο.

Το Πεδιο _id

Το πεδίο _id έχει την ακόλουθη συμπεριφορά και περιορισμούς:

- Από προεπιλογή, το MongoDB δημιουργεί ένα μοναδικό index στο πεδίο _id κατά τη δημιουργία μιας συλλογής.
- Το πεδίο _id είναι πάντα το πρώτο πεδίο στα έγγραφα. Αν ο server λαμβάνει έναέγγραφο που δεν έχει το πεδίο _id πρώτα, τότε ο διακομιστής θα κινηθεί προς το αρχικό πεδίο.
- Το πεδίο _id μπορεί να περιέχει τιμές για κάθε τύπο δεδομένων BSON, εκτός από μια σειρά.

Τα παρακάτω είναι κοινές επιλογές για την αποθήκευση των τιμών για _id:

- Χρησιμοποιήστε ένα ObjectId.
- Χρησιμοποιήστε ένα φυσικό μοναδικό αναγνωριστικό, έαν υπάρχει. Αυτό εξοικονομεί χώρο και αποφεύγει ένα πρόσθετο index.
- Δημιουργήστε έναν αριθμό με αυτόματη προσαύξηση.
- Δημιουργήστε ένα UUID μέσα στον κώδικα της εφαρμογής σας. Για μια πιο αποτελεσματική αποθήκευση των UUID values μέσα στην συλλογή και μέσα στο _id του index, απόθηκέυστε το UUID σαν value του BSON BinData type.

Index keys τα οποία είναι του τύπου BinData είναι πιο αποτελεσματικά στην αποθηκευμένα στο index εάν :

η binary subtype value είναι στο εύρος 0-7 ή 128-135, και

το μήκος της συστοιχίας byte είναι: 0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, ή 32.

Χρησιμοποιήστε την εγκατάσταση του driver σας BSON UUID για να δημιουργήσει UUIDs. Θα πρέπει να γνωρίζετε ότι ο driver μπορεί να εφαρμόσει διαφορετικιά UUID serialization και deserialization logic, η οποία δεν μπορεί να είναι πλήρως συμβατή με τους άλλους driver.

Dot Notation

Η MongoDB χρησιμοποιεί το dot notation για να έχει πρόσβαση στα στοιχεία του πίνακα και να προσφέρει πρόσβαση στα πεδία ενός ενσωματωμένου εγγράφου.Για πρόσβαση σε ένα στοιχείο ενός πίνακα από τη θέση του δείκτη με μηδενική βάση, απαιτείται κατά σειρά το όνομα της συστοιχίας με την τελεία (.) και μηδενική βάση στη θέση του index, και να επισυνάψουν σε εισαγωγικά (.):

```
'<array>.<index>'
```

Δείτε επίσης $ positional operatorς για πράξεις update και $ projection operator όταν συστοιχία της θέση του index είναι άγνωστη.

Για πρόσβαση σε ένα πεδίο ενός ενσωματωμένου εγγράφου με dot-notation, απαιτείται κατά σειρά το ενσωματωμένο όνομα του εγγράφου με την τελεία και το όνομα του πεδίου, και να επισυνάψετε σε εισαγωγικά (.):

```
'<embedded document>.<field>'
```

Μοντέλο Δεδομένων

Τα δεδομένα στο MongoDB έχουν ένα ευέλικτο σχήμα. Σε αντίθεση με τις βάσεις δεδομένων SQL, όπου θα πρέπει να καθορίσουν και να αναγνωρίσουν το σχήμα ενός πίνακα πριν από την εισαγωγή των δεδομένων, οι συλλογές του MongoDB δεν επιβάλλουν την δομή του εγγράφου. Η ευελιξία αυτή διευκολύνει τη χαρτογράφηση των εγγράφων σε μια οντότητα ή αντικείμενο. Κάθε έγγραφο μπορεί να ταιριάξει με τα πεδία δεδομένων του εκπροσωπούμενης οντότητας, ακόμη και αν τα δεδομένα έχουν σημαντική διακύμανση. Στην πράξη, ωστόσο, τα έγγραφα σε μια συλλογή μοιράζονται μια παρόμοια δομή.

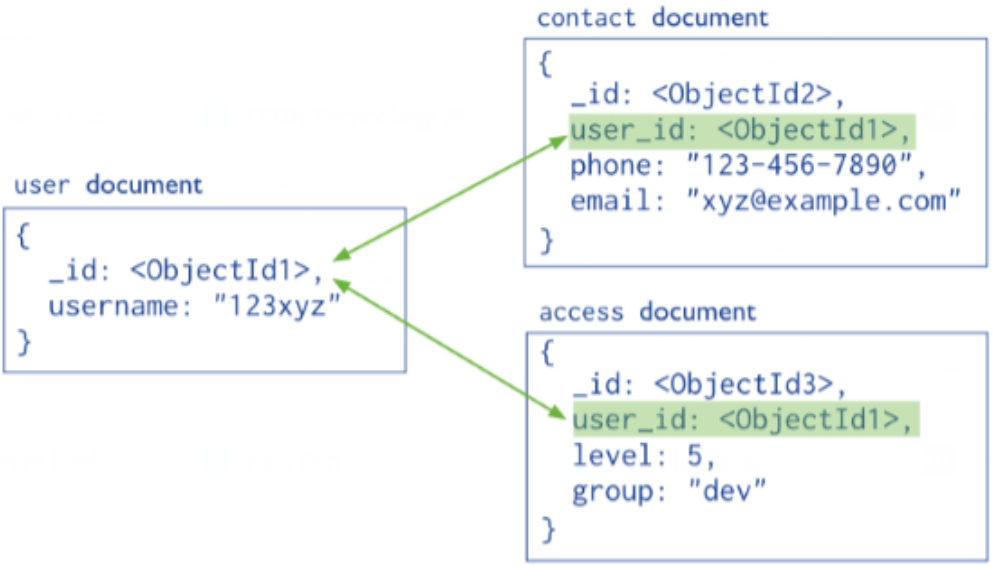
Η βασική πρόκληση για τη μοντελοποίηση δεδομένων ισορροπεί τις ανάγκες της εφαρμογής, τα χαρακτηριστικά της απόδοσης της μηχανής βάσης δεδομένων, και τα μοτίβα ανάκτησης δεδομένων. Κατά τον σχεδιασμό μοντέλων δεδομένων, απαιτείται πάντα η χρήση της εφαρμογής των δεδομένων (δηλαδή queries, updates, και την επεξεργασία των δεδομένων), καθώς και την εγγενή δομή των δεδομένων.

Δομή Εγγράφων

Η βασική απόφαση για το σχεδιασμό μοντέλων δεδομένων για εφαρμογές MongoDB περιστρέφεται γύρω από την δομή των εγγράφων και πώς η εφαρμογή αντιπροσωπεύει τις σχέσεις μεταξύ των δεδομένων. Υπάρχουν δύο εργαλεία που επιτρέπουν στις εφαρμογές να εκπροσωπούν αυτές τις σχέσεις: αναφορές και ενσωματωμένα έγγραφα.

Αναφορές

Τα References αποθηκεύουν τις σχέσεις μεταξύ των στοιχείων με συνδέσμους ή αναφορές από το ένα έγγραφο σε άλλο. Οι αιτήσεις μπορούν να επιλύσουν τις αναφορές αυτές να έχουν πρόσβαση στα συναφή δεδομένα. Σε γενικές γραμμές, αυτές είναι κανονικοποιημένα μοντέλα δεδομένων.



Embedded Data

Τα ενσωματωμένα έγγραφα καταγράφουν τις σχέσεις μεταξύ των δεδομένων από την αποθήκευση των σχετικών δεδομένων σε μια ενιαία δομή του εγγράφου. Τα έγγραφα της MongoDB επιτρέπουν να ενσωματώσετε δομές έγγραφου ως υπο-έγγραφα σε ένα πεδίο ή μια σειρά μέσα σε ένα έγγραφο. Αυτά τα denormalized μοντέλα δεδομένων επιτρέπουν στις εφαρμογές να ανακτούν related data σε μια ενιαία λειτουργία της βάσης δεδομένων.

Atomicity of Write Operations

Στην MongoDB, οι εργασίες εγγραφής είναι ατομικές σε επίπεδο εγγράφου, και καμία λειτουργία εγγραφής δεν μπορεί να επηρεάσει ατομικά περισσότερα από ένα έγγραφο ή περισσότερες από μία συλλογή. Ένα denormalized μοντέλο δεδομένων με ενσωματωμένα δεδομένα συνδυάζει όλα τα συναφή δεδομένα και αντιπροσωπεύομενη οντότητα σε ένα ενιαίο έγγραφο. Αυτό διευκολύνει ατομικά τις δραστηριότητες εγγραφής δεδομένου ότι μια ενιαία λειτουργία εγγραφής μπορεί να εισάγει ή να ενημερώσει τα στοιχεία για μια οντότητα.

Η κανονικοποίηση των δεδομένων θα χωρίσει τα δεδομένων σε πολλές συλλογές και θα απαιτούσε πολλαπλές λειτουργίες εγγραφής που δεν είναι ατομικές συλλογικά.

Document Growth

Ορισμένα Updates, όπως συμπιεσμένα στοιχεία σε έναν πίνακα ή την προσθήκη νέων πεδίων, αυξάνει το μέγεθος ενός εγγράφου. Εάν το μέγεθος του εγγράφου υπερβαίνει το διαθέσιμο χώρο για το εν λόγω έγγραφο, το MongoDB μεταφέρει το έγγραφο στον δίσκο. Το τίμημα της ανάπτυξης μπορεί να επηρεάσει την απόφαση για την normalize ή denormalize των δεδομένων..

Data Use and Performance

Κατά τον σχεδιασμό ενός μοντέλου δεδομενων πρέπει να λαμβάνεται υπόψιν πώς εφαρμογές θα χρησιμοποιούν τη βάση δεδομένων. Για παράδειγμα, αν η εφαρμογή χρησιμοποιεί μόνο πρόσφατα εισαχθεί έγγραφα, πρέπει να ληφθεί υπόψιν να χρησιμοποιήσετε τις Προσαρμοσμένες Συλλογές. Ή αν οι ανάγκες της εφαρμογής να διαβάσει κυρίως τους χειρισμούς σε μια συλλογή, προσθέτοντας δείκτες για να υποστηρίξει κοινές ερωτήσεις μπορεί να βελτιώσει τις επιδόσεις.

Data Model Design

Αποτελεσματικά μοντέλα δεδομένων υποστηρίζουν τις ανάγκες της εφαρμογής. Το βασικό μέλημα για τη δομή των εγγράφων σας είναι η απόφαση να ενσωματωθούν ή να χρησιμποποιηθούν οι αναφορές.

Embedded Data Models

Με την MongoDB, είναι δυνατό να ανσωματωθούν συναφή δεδομένα σε μια ενιαία δομή ή ένα έγγραφο. Αυτά το σχήμα είναι γενικά γνωστό ως “denormalized" μοντέλο, και μπορεί να δώσει σημαντικό πλεονέκτημα μέσω των rich εγγράφων της MongoDB. Ας εξετάσουμε το ακόλουθο διάγραμμα:

Τα ενσωματωμένα μοντέλα δεδομένων επιτρέπουν στις εφαρμογές να αποθηκεύουν συνδεδεμένα κομμάτια πληροφοριών στο ίδιο αρχείο βάσης δεδομένων. Ως αποτέλεσμα, οι εφαρμογές μπορεί να χρειαστούν να εκδώσουν λιγότερα queries και updates για να ολοκληρώσουν κοινές εργασίες.

Σε γενικές γραμμές, τα ενσωματωμένα μοντέλα δεδομένων χρησιμοποιούνται όταν:

- συμπεριλαμβάνονται οι σχέσεις μεταξύ των οντοτήτων.
- υπάρχουν μια-προς-πολλές σχέσεις μεταξύ των οντοτήτων. Σε αυτές τις σχέσεις οι «πολλοί» ή μικρά έγγραφα πάντα εμφανίζονται με ή αντιμετωπίζονται στο πλαίσιο του «ένα» ή γονέα έγγραφα.

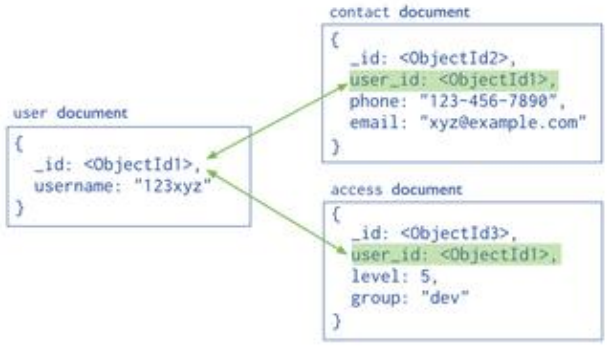
Σε γενικές γραμμές, η ενσωμάτωση παρέχει καλύτερη απόδοση για τις λειτουργίες ανάγνωσης, καθώς και τη δυνατότητα να ζητήσετε και να ανακτήσετε τα συναφή δεδομένα σε μια ενιαία λειτουργία της βάσης δεδομένων. Τα ενσωματωμένα μοντέλα δεδομένων καθιστούν δυνατή την ενημέρωση των σχετικών δεδομένων σε μια ενιαία λειτουργία ατομικής εγγραφής.

Ωστόσο, η ενσωμάτωση των σχετικών δεδομένων σε έγγραφα μπορεί να οδηγήσει σε καταστάσεις όπου τα έγγραφα αυξάνονται μετά τη δημιουργία τους. Η ανάπτυξη του εγγράφου μπορεί να επηρεάσει την απόδοση εγγραφής και να οδηγήσει σε κατακερματισμό των δεδομένων. Επιπλέον, τα έγγραφα στο MongoDB πρέπει να είναι μικρότερα από το μέγιστο μέγεθος εγγράφου BSON.

Για αλληλεπίδραση με τα ενσωματωμένα έγγραφα, είναι απαραίτητη η χρήση dot notation για "reach into" σε ενσωματωμένα έγγραφα.

Normalized Data Models

Τα Normalized μοντέλα δεδομένων περιγράφουν τις σχέσεις με χρήση αναφορών μεταξύ των εγγράφων.



2.3 Couch DB

HCouchDB είναι μια βάση δεδομένων για Web που χρησιμοποιείται ευρέως σε εφαρμογές διαδικτύου. Τα δεδομένα απουθηκεύονται με την μορφη εγγράφων JSON, μέσω του πρωτοκόλλου http. Οι λειτουργίες index, combine και transform επί των εγγράφων

Η CouchDB είναι μια βάση δεδομένων που αγκαλιάζει πλήρως το διαδίκτυο. Αποθηκεύστε τα δεδομένα σας με έγγραφα τύπου JSON. Δείτε τα έγγραφά σας και να διερευνούν δείκτες σας με το web browser σας, μέσω HTTP. Index, combine, and transform τα έγγραφά σας με JavaScript. Η CouchDB λειτουργεί καλά με σύγχρονες web και mobile εφαρμογές. Μπορεί να εξυπηρετήσει ακόμα και web εφαρμογές. Και μπορείτε να διανείμετε τα δεδομένα σας, ή τις εφαρμογές σας, αποτελεσματικά, χρησιμοποιώντας την σταδιακή αντιγραφή του CouchDB. Η CouchDB υποστηρίζει master-master ρυθμίσεις με αυτόματη ανίχνευση συγκρούσεων.

Η CouchDB έρχεται με μια σειρά από χαρακτηριστικά, όπως η on-the-fly μετατροπή των εγγράφων και των κοινοποιήσεων με αλλαγή σε πραγματικό χρόνο, που κάνει την ανάπτυξη των web εφαρμογών εύκολη. Διαθετει επισης μια εύκολη κονσόλα στη χρήση του web administration. Η CouchDB είναι διαθέσιμη σε μεγάλο βαθμό καθώς και partition tolerant, αλλά είναι επίσης συνεπής. Η CouchDB έχει ένα ανεκτικό ελάττωμα, την μηχανή αποθήκευσης που βάζει πρώτα την ασφάλεια των δεδομένων σας.

Χαρακτηρίζεται ως μια «NoSQL" βάση δεδομένων, ένας όρος που έγινε όλο και πιο δημοφιλής στα τέλη του 2009, και στις αρχές του 2010. Αν και αυτός ο όρος είναι μάλλον ο γενικός χαρακτηρισμός μιας βάσης δεδομένων, ή αποθηκευμένων δεδομένων, καθορίζεται με σαφήνεια ένα διάλειμμα από τις παραδοσιακές SQL - based βάσεις δεδομένων. Μια βάση δεδομένων σH CouchDB στερείται ενός σχήματος, ή άκαμπτων δομών των προκαθορισμένων δεδομένων, όπως οι πίνακες. Τα δεδομένα που αποθηκεύονται στην CouchDB είναι έγγραφα (-α) JSON. Η δομή των δεδομένων, ή του έγγραφου (-ων), μπορεί να αλλάξει δυναμικά για να φιλοξενήσει τις εξελισσόμενες ανάγκες.

What it is Not

Για να κατανοήσουμε καλύτερα τι είναι η CouchDB, είναι χρήσιμο να προτίστως να δούμε τι δεν είναι η CouchDB:

- Μια σχεσιακή βάση δεδομένων. Οι διαφορές αυτές αρθρώνονται πάνω στην ενότητα Meet CouchDB, και άλλα τμήματα του Wiki.
- Η αντικατάσταση όλων των βάσεων δεδομένων. Κατά την ανάπτυξη και το σχεδιασμό ένα καλό σύστημα πληροφόρησης στο οποίο θα πρέπει να επιλέξετε το καλύτερο εργαλείο για τη δουλειά. Ενώ Η CouchDB μπορεί να χρησιμοποιηθεί σε μια ευρεία ποικιλία εφαρμογών που μπορείτε να διαπιστώσετε ότι ένα άλλο data store είναι ταιριάζει καλύτερα για το πρόβλημά σας. Εάν είστε νέοι σH CouchDB, και δεν είσται σίγουρος αν αυτό είναι κατάλληλο για το πρόβλημα της διαχείρισης των δεδομένων σας, ρωτήστε τους άλλους για τη λίστα και την #couchdb κανάλι IRC για συμβουλές.

- Μια object-oriented βάση δεδομένων. Ενώ τα CouchDB καταστήματα JSON αντικείμενα, δεν έχουν ως στόχο να λειτουργήσουν ως ένα επίμονο ομοιογενές στρώμα για μια αντικειμενοστραφή γλώσσα προγραμματισμού.

Βασικά Χαρακτηριστικά

Documents

Ένα έγγραφο CouchDB είναι ένα αντικείμενο JSON που αποτελείται από ονομαστηκά πεδία. Οι τιμές των πεδίων μπορεί να είναι strings, αριθμοί, ημερομηνίες, ή ακόμη ordered lists και associative maps. Ένα παράδειγμα ενός εγγράφου θα είναι ένα blog post:

```
{
    "Subject": "I like Plankton",
    "Author": "Rusty",
    "PostedDate": "5/23/2006",
    "Tags": ["plankton", "baseball", "decisions"],
    "Body": "I decided today that I don't like baseball. I like plankton."
}
```

Στο παραπάνω παράδειγμα εγγράφου, Το Θέμα είναι ένα πεδίο που περιέχει ένα μόνο string value "I like plankton". Τα tags είναι ένα πεδίο που περιέχει τον κατάλογο των τιμών "plankton", "baseball", και "decisions".

Μια βάση δεδομένων CouchDB είναι μια επίπεδη συλλογή των εν λόγω εγγράφων. Κάθε έγγραφο που αναγνωρίζεται από ένα μοναδικό αναγνωριστικό.

Views

Για να αντιμετωπιστεί αυτό το πρόβλημα της προσθήκης δομής στα ημι-δομημένα δεδομένα, η CouchDB ενσωματώνει ένα view model χρησιμοποιώντας JavaScript για περιγραφή. Views είναι η μέθοδος της άθροισης και η υποβολή εκθέσεων σχετικά με τα έγγραφα σε μια βάση δεδομένων, και είναι χτισμένα on-demand να συσσωματώνονται, να συμμετάσχουν και να υποβάλουν έκθεση σχετικά με τα έγγραφα της βάσης δεδομένων. Views που χτίζονται δυναμικά και δεν επηρεάζουν την υποκείμενο έγγραφο. Είναι δυνατή η τήρηση πολλών διαφορετικών Views. Στοιχειώδη updates στα έγγραφα δεν απαιτούν την πλήρη αποκατάσταση re-indexing των views.

Schema-Free

Σε αντίθεση με τις βάσεις δεδομένων SQL, οι οποίες έχουν σχεδιαστεί για να αποθηκεύουν και να υποβάλουν report εξαιρετικά δομημένο, αλληλένδετα δεδομένα, Η CouchDB έχει σχεδιαστεί για να αποθηκεύει και να υποβάλει έκθεση σχετικά με τα μεγάλα ποσά των ημι-δομημένων, document-oriented δεδομένων. Η CouchDB απλοποιεί σε μεγάλο βαθμό την ανάπτυξη του εγγράφου με γνώμονα τις εφαρμογές του, όπως της συνεργατικές εφαρμογές web.

Σε μια βάση δεδομένων SQL, το σχήμα και η αποθήκευση των υφιστάμενων δεδομένων πρέπει να ενημερώνεται ανάλογα με τις ανάγκες που εξελίσσονται. Με την CouchDB, δεν απαιτείται σχήμα, έτσι ώστε οι νέοι τύποι εγγράφων με νέο νόημα μπορούν να προστεθούν με ασφάλεια παράλληλα με τα παλιά. Ωστόσο, για εφαρμογές που απαιτούν ισχυρή επικύρωση των νέων εγγράφων είναι δυνατόν οι προσαρμοσμένες λειτουργίες επικύρωσης. Η view engine έχει σχεδιαστεί για να χειριστεί εύκολα νέους τύπους εγγράφων.

Distributed

Η CouchDB είναι ένα peer based κατανεμημένο σύστημα βάσης δεδομένων. Οποιοσδήποτε αριθμός των CouchDB hosts (servers και offline-clients) μπορούν να έχουν ανεξάρτητα "αντίγραφα ρέπλικες» στην ίδια βάση δεδομένων, όπου οι εφαρμογές έχουν πλήρη database διαδραστικότητα (query, add, edit, delete). Κατά την επιστροφή στο διαδίκτυο ή σε ένα χρονοδιάγραμμα, οι αλλαγές στις βάσεις δεδομένων μπορούν να αναπαραχθούν αμφίδρομα. Η CouchDB έχει ενσωματωμένη ανίχνευση και τη διαχείριση των συγκρούσεων και η διαδικασία αναπαραγωγής είναι οριακή και γρήγορη, αντιγράφοντας μόνο έγγραφα που έχουν αλλάξει από την προηγούμενη αντιγραφή. Οι περισσότερες εφαρμογές δεν απαιτούν ειδικό σχεδιασμό για να επωφεληθούν των διανεμόμενων ενημερώσεων και αντιγραφών. Σε αντίθεση με επαχθείς προσπάθειες για να μπουλόνι διανέμονται χαρακτηριστικά πάνω από τα ίδια μοντέλα κληρονομιά και βάσεις δεδομένων, αντιγραφής σε CouchDB είναι το αποτέλεσμα προσεκτικής εδάφους-up σχεδιασμού, της μηχανικής και της ολοκλήρωσης. Το πλαίσιο αυτό αντιγραφής παρέχει ένα ολοκληρωμένο σύνολο χαρακτηριστικών:

- Master → Slave replication
- Master ↔ Master replication
- Filtered Replication
- Incremental and bi-directional replication
- Conflict management

Αυτές οι λειτουργίες αναπαραγωγής μπορεί να χρησιμοποιηθούν σε συνδυασμό για να δημιουργήσουν ισχυρές λύσεις σε πολλά προβλήματα στον κλάδο της πληροφορικής. Εκτός από τα φανταστικά χαρακτηριστικά στην αντιγραφή, την αξιοπιστία και την επεκτασιμότητα του CouchDB ενισχύεται περαιτέρω με το να εφαρμόζονται στη γλώσσα προγραμματισμού Erlang. Η Erlang έχει ενσωματωμένη

υποστήριξη για συναγωνισμό, τη διανομή, την ανοχή σφαλμάτων, και έχει χρησιμοποιηθεί για χρόνια για να δημιουργηθούν αξιόπιστα συστήματα στον κλάδο των τηλεπικοινωνιών. Από τη σχεδίαση, τη γλώσσα Erlang και του χρόνου εκτέλεσης είναι σε θέση να επωφεληθεί από τα νεότερα hardware με πολλαπλούς πυρήνες CPU.

Document Storage

Ένας server του CouchDB φιλοξενεί ονομαστικά βάσεις δεδομένων, οι οποίες αποθηκεύουν ** έγγραφα **. Κάθε έγγραφο έχει το δικό του όνομα στη βάση δεδομένων, και Η CouchDB παρέχει μια ξεκούραστη HTTP API για την ανάγνωση και την ενημέρωση (προσθήκη, επεξεργασία, διαγραφή) στα έγγραφα της βάσης δεδομένων.

Τα έγγραφα αποτελούν την κύρια μονάδα δεδομένων σΗ CouchDB και αποτελούνται από οποιοδήποτε αριθμό των πεδίων και τα συνημμένων. Τα έγγραφα περιλαμβάνουν επίσης metadata που είναι συντηρημένα από το σύστημα της βάσης δεδομένων. Τα πεδία των έγγραφων το δικό του όνομα και περιέχουν τιμές των διαφόρων τύπων (κείμενο, αριθμός, boolean, κατάλογοι, κ.λπ.), και δεν υπάρχει καθορισμένο όριο για το μέγεθος του κειμένου ή τον αριθμό των στοιχείων.

Το μοντέλο ενημέρωσης του CouchDB εγγράφου είναι lockless και optimistic. Οι αλλαγές στο έγγραφο γίνονται από client applications φορτώνοντας τα έγγραφα, την εφαρμογή των αλλαγών, και αποθηκεύοντας τα πίσω στη βάση δεδομένων. Εάν κάποιος άλλος πελάτης επεξεργάζεστε το ίδιο έγγραφο εξοικονομεί πρώτες αλλαγές τους, ο πελάτης λαμβάνει ένα σφάλμα σύγκρουση επεξεργασία κατά την αποθήκευση. Για την επίλυση της σύγκρουσης ενημερωμένη έκδοση, η τελευταία έκδοση του εγγράφου μπορεί να ανοίξει, οι αλλαγές ξαναχρησιμοποιηθεί και η ενημέρωση προσπάθησε και πάλι.

Updates του έγγραφου (add, edit, delete) είναι όλα ή τίποτα, είτε πετυχαίνοντας εξ ολοκλήρου ή αποτυγχανοντας τελείως. Η βάση δεδομένων δεν περιέχει μερικώς αποθηκευμένα ή επεξεργασμένα έγγραφα.

Acid Properties

Η διάταξη αρχείων σΗ CouchDB και η δέσμευση του συστήματος ότι διαθέτει όλες τις Atomic Consistent Isolated Durable (ACID) ιδιότητες. Στον δίσκο, Η CouchDB ποτέ δεν αντικαθιστά δεσμευμένα δεδομένα ή σχετικές δομές, εξασφαλίζοντας ότι το αρχείο βάσης δεδομένων είναι πάντα σε σταθερή κατάσταση. Αυτό είναι ένα «crash-only", εκεί όπου ο διακομιστής του CouchDB δεν περνά από μια διαδικασία τερματισμού λειτουργίας, είναι απλά τερματισμένος.

Τα Updates του έγγραφου (add, edit, delete) είναι με σειριακούς αριθμούς, εκτός από τις δυαδικές άμορφες μάζες που είναι γραμμένες ταυτόχρονα. Οι αναγνώστες της βάσης δεδομένων δεν είναι ποτέ κλειδωμένοι έξω και δεν χρειάζονται να περιμένουν για συγγραφείς ή άλλους αναγνώστες. Οποιοσδήποτε αριθμός των πελατών μπορεί να κάνει ανάγνωση των εγγράφων, χωρίς να είναι κλειδωμένα ή να διακόπτονται από ταυτόχρονες ενημερώσεις, ακόμα και στο ίδιο έγγραφο. Οι CouchDB λειτουργίες ανάγνωσης χρησιμοποιούν ένα Multi-Version Concurrency Control (MVCC) μοντέλο, όπου κάθε πελάτης βλέπει ένα συνεπές στιγμιότυπο της βάσης δεδομένων από την αρχή μέχρι το τέλος της λειτουργίας της ανάγνωσης.

Τα έγγραφα αναπροσαρμόζονται σε B-trees με το όνομά τους (DocID) και ένα Sequence ID. Κάθε ενημέρωση σε ένα στιγμιότυπο της βάσης δεδομένων δημιουργεί ένα νέο αύξοντα αριθμό. Τα Sequence IDs χρησιμοποιούνται αργότερα για την εύρεση αυξητικών αλλαγών σε μια βάση δεδομένων. Αυτά τα B-trees index ενημερώνονται ταυτόχρονα όταν τα έγγραφα αποθηκεύονται ή διαγράφονται. Οι ενημερώσεις δείκτης πάντα συμβαίνουν στο τέλος του αρχείου (append μόνο ενημερώσεις).

Τα έγγραφα έχουν το πλεονέκτημα των δεδομένων που έχουν ήδη βολική συσκευασία για αποθήκευση και όχι χωρισμένη σε πολυάριθμους πίνακες και σειρές στα περισσότερα συστήματα βάσεων δεδομένων. Όταν τα έγγραφα έχουν δεσμευθεί στον δίσκο, τα πεδία των εγγράφων και των μεταδιδόμενων που συσκευάζονται σε ρυθμιστικά, διαδοχικά το ένα μετά το άλλο έγγραφο (χρήσιμο αργότερα για την αποτελεσματική οικοδόμηση απόψεων).

Όταν τα έγγραφα του CouchDB ενημερώνονται, όλα τα δεδομένα και συναφείς δείκτες ξεπλένονται στον δίσκο και η **συναλλακτική** δεσμεύονται πάντα αφήνει τη βάση δεδομένων σε μια εντελώς σταθερή κατάσταση. Δεσμεύεται να συμβεί σε δύο στάδια:

- Όλα τα δεδομένα του εγγράφου και των σχετικών ενημερώσεων των δεικτών ξεπλένονται συγχρονισμένα στον δίσκο.
- Η ενημερωμένη κεφαλίδα της βάσης δεδομένων είναι γραμμένη σε δύο διαδοχικά, πανομοιότυπα κομμάτια για να δημιουργήσουν τα πρώτα 4κ του αρχείου, και στη συνέχεια συγχρονισμένα ξεπλένονται στο δίσκο.

Σε περίπτωση συντριβής του OS ή διακοπής του ρεύματος κατά τη διάρκεια του βήματος 1, οι μερικώς ξεπλένόμενες ενημερώσεις απλά ξεχνίουνται κατά την επανεκκίνηση. Αν μια τέτοια σύγκρουση συμβεί κατά τη διάρκεια του σταδίου 2 (τη διάπραξη του header), το σωζόμενο αντίγραφο των προηγούμενων πανομοιότυπων header θα παραμείνει, διασφαλίζοντας τη συνοχή όλων των δεδομένων που είχαν προηγουμένως δεσμευτεί.. Με εξαίρεση την περιοχή του header, οι ελέγχει συνέπειας ή fix-ups μετά από μια συντριβή ή μια διακοπή ρεύματος δεν είναι ποτέ αναγκαίοι.

Compaction

Η σπατάλη χώρου ανακτάται από μια περιστασιακή συμπίεση. Σύμφωνα με το χρονοδιάγραμμα, όταν το αρχείο βάσης δεδομένων υπερβαίνει ένα ορισμένο ποσό του ανεκμετάλλευτου χώρου, η διαδικασία συμπίεσης κλωνοποιεί όλα τα ενεργά δεδομένα σε ένα νέο αρχείο και, στη συνέχεια, απορρίπτει το παλιό αρχείο. Η βάση δεδομένων παραμένει εντελώς on-line στο σύνολο του χρόνου και όλες οι ενημερωμένες εκδόσεις και οι αναγνώσεις επιτρέπουν να ολοκληρωθεί με επιτυχία. Το παλιό αρχείο διαγράφεται μόνο όταν όλα τα στοιχεία του έχουν αντιγραφεί και όλοι οι χρήστες μεταφέρονται στο νέο αρχείο.

Views

Οι ACID ιδιότητες ασχολούνται μόνο με την αποθήκευση και τα updates, χρειαζόμαστε επίσης τη δυνατότητα να εμφανίζονται τα δεδομένα μας με ενδιαφέρον και χρήσιμους τρόπους. Σε αντίθεση με άλλες SQL βάσεις δεδομένων

όπου τα δεδομένα θα πρέπει να αναλυθούν προσεκτικά σε πίνακες, τα στοιχεία σΗ CouchDB αποθηκεύονται σε ημι-δομημένα εγγράφα. Τα έγγραφα του CouchDB είναι ευέλικτα και το καθένα έχει τη δική του σιωπηρή δομή, η οποία ανακουφίζει τα πιο δύσκολα προβλήματα και τις παγίδες των αμφίδρομων αναπαραγωγημένων σχηματικών πινάκων και, των περιλαμβανομένων δεδομένων τους.

Αλλά πέρα από ότι ενεργεί ως έναν φανταχτερό server αρχείων, ένα απλό μοντέλο έγγραφων για την αποθήκευση δεδομένων και την ανταλλαγή είναι πολύ απλό για να οικοδομήσουμε πραγματικές εφαρμογές - απλά δεν κάνουν αρκετά από τα πράγματα που θέλουμε και περιμένουμε. Θέλουμε να ρίξουμε τα ζάρια και να δείτε τα δεδομένα μας με πολλούς διαφορετικούς τρόπους. Αυτό που χρειάζεται είναι ένας τρόπος για να φιλτράρει, να οργανώσει και να υποβάλει έκθεση σχετικά με τα δεδομένα που δεν έχει αναλυθεί σε πίνακες.

View Model

Για να αντιμετωπιστεί αυτό το πρόβλημα της προσθήκης δομής πίσω στα αδόμητα και ημι-δομημένα δεδομένα, Η CouchDB ενσωματώνει ένα view model. Τα views είναι η μέθοδος της άθροισης και της υποβολής εκθέσεων σχετικά με τα έγγραφα σε μια βάση δεδομένων, και είναι χτισμένα on-demand για τη συγκέντρωση, την συμμετοχή και την υποβολή μιας έκθεσης σχετικά με τα έγγραφα της βάσης δεδομένων. Τα views χτίζονται δυναμικά και δεν επηρεάζουν το υποκείμενο έγγραφο, μπορείτε να έχετε όσες διαφορετικές απόψεις των αναπαραστάσεων των ίδιων δεδομένων όπως σας αρέσει.

Οι ορισμοι των views είναι απολύτως εικονικοί και εμφανίζονται μόνο τα έγγραφα από το τρέχον στιγμιότυπο της βάσης δεδομένων, καθιστώντας τα να διαχωρισθούν από τα στοιχεία που εμφανίζουν και συμβατά με αναπαραγωγή. Τα views του CouchDB ορίζονται μέσα σε ειδικά ** έγγραφα σχεδιασμού ** και μπορούν να αναπαραχθούν σε όλες τις εμφανίσεις της βάσης δεδομένων, όπως τα κανονικά έγγραφα, έτσι ώστε όχι μόνο τα δεδομένα να αντιγράφονται σΗ CouchDB, αλλά ολόκληρα σχέδια εφαρμογής αναπαράγονται επίσης.

Javascript View Functions

Τα views που ορίζονται χρησιμοποιώντας τις λειτουργίες της Javascript ενεργούν ως μέρος του χάρτη σε ένα μειωμένο σύστημα χαρτογράφησης. Μια λειτουργία του view λαμβάνει ένα έγγραφο του CouchDB ως επιχείρημα και, στη συνέχεια, κάνει ότι υπολογισμό πρέπει να κάνει για να καθορίσει τα δεδομένα που πρόκειται να διατεθούν μέσω του view, αν υπάρχουν. Αυτό μπορεί να προσθέσει πολλές σειρές στο view based σε ένα ενιαίο έγγραφο, ή μπορεί να μην προσθέσει καθόλου.

View Indexes

Τα views είναι μια δυναμική αναπαράσταση του πραγματικού περιεχομένου του εγγράφου της βάσης δεδομένων, και Η CouchDB καθιστά εύκολο να δημιουργήσετε χρήσιμα views της βάσης δεδομένων. Αλλά δημιουργώντας ένα view μιας βάσης δεδομένων με εκατοντάδες χιλιάδες ή εκατομμύρια έγγραφα είναι χρονοβόρο και δαπανηρό, δεν είναι κάτι που το σύστημα πρέπει να κάνει από την αρχή κάθε φορά.

Για να διατηρήσετε το view querying γρήγορο, η μηχανή του view υποστηρίζει ευρετήρια με τα views, και σταδιακά να τα ενημερώνει ώστε να αντικατοπτρίζει τις αλλαγές στη βάση δεδομένων. Ο σχεδιασμός του πυρήνα του CouchDB είναι σε μεγάλο βαθμό βελτιστοποιημένος γύρω από την ανάγκη για αποδοτική, σταδιακή δημιουργία των views και των ευρετηρίων τους.Τα views και οι λειτουργίες τους που ορίζονται στο εσωτερικό ειδικό «σχέδιο» των έγγραφων, καθώς και ένα έγγραφο σχεδιασμού μπορεί να περιέχει οποιοδήποτε αριθμό με δικές του ονομαστικές λειτουργίες των views. Όταν ένας χρήστης ανοίγει ένα view και ο δείκτης του ενημερώνεται αυτόματα, όλες τα views στο ίδιο έγγραφο σχεδιασμού αναπροσαρμόζονται ως μία ενιαία ομάδα.Ο view builder χρησιμοποιεί το sequence ID της βάσης δεδομένων για να καθοριστεί αν η view ομάδα είναι πλήρως up-to-date με τη βάση δεδομένων. Αν όχι, το view engine εξετάζει σε όλα τα έγγραφα της βάσης δεδομένων (συσκευασμένα σε διαδοχική σειρά) αλαγμένα από την τελευταία ανανέωση. Τα έγγραφα που διαβάζονται με τη σειρά που εμφανίζονται στο αρχείο στον δίσκο, επιδιώκει να μειώσει την συχνότητα και το κόστος της κεφαλή του δίσκου.Τα views μπορούν να διαβάσουν και queried ταυτόχρονα, καθώς ανανεώνετε. Εάν ένας πελάτης έχει αργή ροή στα περιεχόμενα ενός μεγάλου view, το ίδιο view μπορεί να ανοίξει ταυτόχρονα και ανανεώνεται για άλλον πελάτη χωρίς κλείδωμα του πρώτου πελάτη. Αυτό ισχύει για οποιοδήποτε αριθμό των ταυτόχρονων αναγνώστες πελάτη, ο οποίος μπορεί να διαβάσει και να διερευνούν τη θέα, ενώ ο δείκτης ταυτόχρονα να ανανεώνεται για άλλους πελάτες χωρίς να προκαλεί προβλήματα για τους αναγνώστες.Καθώς τα έγγραφα που εξετάσθηκαν, οι προηγούμενες τιμές σειρά τους να αφαιρούνται από τους δείκτες άποψη, εφόσον υπάρχουν. Αν έχει επιλεγεί το έγγραφο από τη λειτουργία view, τα αποτελέσματα της λειτουργίας εισάγεται στο view ως νέα σειρά.Όταν οι αλλαγές στο view index γραφτεί στο δίσκο, τα updates είναι πάντα επισυναπτόμενα στο τέλος του αρχείου, εξυπηρετώντας τόσο για τη μείωση του χρόνου αναζήτησης της κεφαλής του δίσκου κατά τη διάρκεια δέσμευσης του δίσκου και για την εξασφάλιση συντριβών και τις διακοπές ρεύματος δεν μπορεί να προκαλέσει καταστροφή των index. Εάν ένα ατύχημα συμβεί κατά την ενημέρωση του view index, οι ελλιπείς ενημερώσεις του index είναι απλά χάμενα και ανοικοδομήμενα σταδιακά από προηγούμενη δεσμευμένη κατάσταση.

Securtiy and Validation

Για την προστασία που μπορεί να διαβάσει και να επικαιροποιεί τα έγγραφα, Η CouchDB έχει μια απλή πρόσβαση ανάγνωσης και ανανέωσης του μοντέλου επικύρωσης που μπορεί να επεκταθεί και να εφαρμόσει προσαρμοσμένα μοντέλα ασφάλειας.

Administrator Access

Η CouchDB βάση δεδομένων εμφανίσεις έχουν λογαριασμούς διαχειριστή. Ο λογαριασμός του διαχειριστή μπορεί να δημιουργήσει άλλους λογαριασμούς διαχειριστή και να ενημέρωσει τα έγγραφα σχεδιασμού. Τα έγγραφα σχεδιασμού είναι ειδικά έγγραφα που περιέχουν τους ορισμούς άποψη και άλλων ειδικών τύπων, καθώς και τακτικές πεδίων και blobs.

Update Validation

Ως έγγραφα τα οποία είναι γραμμένα στον δίσκο, μπορούν να επικυρωθούν δυναμικά με τις λειτουργίες τις JavaScript τόσο για την ασφάλεια και την επικύρωση των δεδομένων. Όταν το έγγραφο που περνά όλα τα κριτήρια επικύρωσης, η ενημέρωση αφήνεται να συνεχιστεί. Εάν η επικύρωση αποτύχει, η ενημέρωση διακόπτεται και ο πελάτης-χρήστης παίρνει μια απάντηση σφάλματος.

Και οι δύο πιστοποιήσεις του χρήστη και το επικαιροποιημένου έγγραφου δίνονται ως είσοδοι στον τύπο επικύρωσης, και μπορεί να χρησιμοποιηθεί για την εφαρμογή προσαρμοσμένου μοντέλου ασφάλειας με την επικύρωση των δικαιωμάτων ενός χρήστη για να ενημερώσει ένα έγγραφο.

Ένας βασικός "author only" εξημερωμένο μοντέλο έγγραφου είναι ασήμαντο για την υλοποίηση, όπου το έγγραφο ενημερώνεται ειναι επικυρώμενοι για να ελέγξει αν ο χρήστης είναι εισηγμένος στο «author» πεδίο στο υπάρχον έγγραφο. Επίσης είναι δυνατόν να υπάρχουν πιο δυναμικά μοντέλα, όπως ο έλεγχος ενός ξεχωριστού προφίλ του λογαριασμού του χρήστη για τις ρυθμίσεις αδειών.

Οι ενημερωμένες επικυρώσεις επιβάλλονται τόσο στην ζωντανή χρήση όπως και να αναπαραχθούν ενημερώσεις, εξασφαλίζοντας την επικύρωση της ασφάλειας και των δεδομένων σε ένα κοινό, κατανεμημένο σύστημα.

Distributed Updates and Replication

Η CouchDB είναι ένα peer-based κατανεμημένο σύστημα βάσεων δεδομένων, επιτρέπει στους χρήστες και τους διακομιστές να έχουν πρόσβαση και να ενημερώνουν τα ίδια κοινόχρηστα δεδομένα, ενώ αποσυνδέεται και στη συνέχεια αμφίδρομα αναπαράγουν αυτές τις αλλαγές αργότερα.

Η αποθήκευση εγγράφων στην CouchDB, του view και των μοντέλων ασφάλειας είναι σχεδιασμένα να εργάζονται μαζί για να κάνουν πραγματική την αμφίδρομη αντιγραφή, καθώς και αποτελεσματική και αξιόπιστη. Και τα έγγραφα καθώς και τα σχέδια μπορούν να αναπαράγονται, επιτρέποντας την πλήρη εφαρμογή της βάσής δεδομένων (συμπεριλαμβανομένων των εφαρμογών του σχεδιασμού, της λογικής και των δεδομένων) για να αναπαραχθούν σε φορητούς υπολογιστές για χρήση χωρίς σύνδεση, ή να αναπαραχθούν σε servers σε απομακρυσμένα γραφεία όπου οι αργές ή αναξιόπιστες συνδέσεις για να μοιράζονται με δυσκολία τα δεδομένα.

Η διαδικασία αναπαραγωγής είναι οριακή. Σε επίπεδο βάσης δεδομένων, η αντιγραφή εξετάζει μόνο τα έγγραφα που έχουν ενημερωθεί μετά την τελευταία αντιγραφή. Στη συνέχεια, για κάθε επικαιροποιημένο έγγραφο, μόνο τα πεδία και τα blobs που άλλαξαν αναπαράγονται σε όλο το δίκτυο. Αν η αναπαραγωγή αποτύχει σε οποιοδήποτε στάδιο, λόγω προβλημάτων του δικτύου ή συντριβής, για παράδειγμα, στην επόμενη αναπαραγωγή στην επανακίνητε στο ίδιο έγγραφο όπου σταμάτησε.

Τα μερικά αντίγραφα μπορούν να δημιουργηθούν και να διατηρηθούν. Η αναπαραγωγή μπορεί να φιλτράρεται από μια λειτουργία σε JavaScript, έτσι ώστε να αναπαράγονται μόνο συγκεκριμένα έγγραφα ή σε αυτούς που πληρούν συγκεκριμένα κριτήρια. Αυτό μπορεί να επιτρέψει στους χρήστες να λαμβάνουν υποσύνολα έναν ευρύ κοινό offline εφαρμογή βάσης δεδομένων για δική τους χρήση, διατηρώντας ταυτόχρονα φυσιολογική αλληλεπίδραση με την εφαρμογή και αυτού του υποσυνόλου των δεδομένων.

Συγκρούσεις

Οι συγκρούσεις ανίχνευσης και διαχείρισης είναι βασικά θέματα για κάθε κατανεμημένο σύστημα επεξεργασίας. Το σύστημα αποθήκευσης του CouchDB αντιμετωπίζει τις συγκρούσεις επεξεργασίας ως κοινό κράτος, και δεν αποτελούν εξαίρεση. Το μοντέλο χειρισμού των συγκρούσεων είναι απλό και «μη-καταστροφικό», διατηρώντας παράλληλα τις σημασιολογικές του ενιαίου εγγράφου και επιτρέπει την αποκεντρωμένη επίλυση των συγκρούσεων.

Η CouchDB επιτρέπει για οποιοδήποτε αριθμό των αντικρουόμενων εγγράφων που υπάρχουν ταυτόχρονα στη βάση δεδομένων, με κάθε στιγμιότυπο της βάσης δεδομένων νομοτελειακά ν’ αποφασιστεί ποιο έγγραφο είναι ο «νικητής» και ποια

έγγραφα είναι οι συγκρούσεις. Μόνο η νίκη του εγγράφου μπορεί να εμφανιστεί σε views, ενώ "χάνοντας" οι συγκρούσεις εξακολουθούν να είναι προσβάσιμες και παραμένουν στη βάση δεδομένων μέχρι διαγραφούν ή να καθαρίζονται κατά τη συμπύκνωση της βάσης δεδομένων. Επειδή τα έγγραφα σύγκρουσης είναι τακτικά έγγραφα, που αναπαράγονται ακριβώς όπως το κανονικά έγγραφα και υπόκεινται στους ίδιους κανόνες ασφαλείας και επικύρωσης.

Όταν διανέμονται επεξεργασμένες συγκρούσεις, κάθε αντίγραφο της βάσης δεδομένων βλέπει την ίδια νικητήρια αναθεώρηση και το καθένα έχει την ευκαιρία να επιλύσει τη σύγκρουση. Η επίλυση των συγκρούσεων μπορεί να γίνει με το χέρι ή, ανάλογα με τη φύση των δεδομένων και την σύγκρουση, με αυτοματοποιημένους παράγοντες. Το σύστημα καθιστά δυνατή την αποκεντρωμένη επίλυση των συγκρούσεων, διατηρώντας παράλληλα ενιαία την σημασιολογία της βάσης δεδομένων του εγγράφου.

Η διαχείριση συγκρούσεων συνεχίζει να λειτουργεί ακόμη και αν πολλαπλοί αποσυνδεμένοι χρήστες ή παράγοντες επιχειρούν να επιλύσουν τις ίδιες συγκρούσεις. Εάν επιλυθούν οι συγκρούσεις οδηγούν σε μεγαλύτερες συγκρούσεις, το σύστημα τις φιλοξενεί με τον ίδιο τρόπο, προσδιορίζοντας τον ίδιο νικητή σε κάθε μηχανή και τη διατήρηση ενός ενιαίου σημασιολογικού έγγραφου.

Applications

Χρησιμοποιώντας μόνο το βασικό μοντέλο αντιγραφής, πολλές παραδοσιακές εφαρμογές μιας ενιαίας βάσης δεδομένων του διακομιστή μπορεί ναι διανέμεται με σχεδόν καμία επιπλέον εργασία. Η CouchDB έχει σχεδιαστεί για να είναι άμεσα χρήσιμη για βασικές εφαρμογές της βάσης δεδομένων, ενώ επίσης είναι να παραταθεί για πιο περίτεχνα και με πλήρεις δυνατότητες χρήσεις.

Με πολύ λίγη δουλειά της βάσης δεδομένων, είναι δυνατόν να χτίσει μια κατανεμημένη εφαρμογή διαχείρισης εγγράφων με κοκκώδη ασφάλεια και την πλήρη αναθεώρηση του ιστορικού. Οι ενημερώσεις στα έγγραφα μπορούν να εφαρμοστούν για να εκμεταλλευτούν σταδιακά την αναπαραγωγή τομέα και blob, όπου αναπαραχθεί ενημερώσεις είναι σχεδόν εξίσου αποτελεσματικές και στοιχειώδεις ως τα πραγματικά edit differences ("diffs").

Το μοντέλο αντιγραφής του CouchDB μπορεί να τροποποιηθεί για άλλα κατανεμημένα μοντέλα ενημέρωσης. Αν η μηχανή αποθήκευσης που θα επιτρέπει την επικαιροποίηση των συναλλαγών σε πολλαπλά έγγραφα, είναι δυνατό να εκτελέσει Subversion-like “all or nothing” ατομικών δεσμεύσεων όταν αναπαράγει με έναν upstream server, έτσι ώστε οποιαδήποτε μεμονωμένη σύγκρουση του έγγραφου ή η αποτυχία επικύρωσης του θα προκαλέσει ολόκληρη την ενημερωμένη έκδοση να αποτύχει. Όπως η ανατροπή, η σύγκρουση θα μπορούσε να επιλυθεί κάνοντας μια "pull" αντιγραφή για να αναγκάσει τις συγκρούσεις να

είναι σε τοπικό επίπεδο, στη συνέχεια, τη συγχώνευετε και την εκ νέου αναπαράγετε με τον upstream server.

Implementation

Η CouchDB είναι χτισμένη στην πλατφόρμα Erlang OTP, μια λειτουργική, σύγχρονη γλώσσα προγραμματισμού και αναπτυσσόμενης πλατφόρμας. Η Erlang αναπτύχθηκε για εφαρμογές τηλεπικοινωνιών σε πραγματικό χρόνο με εξαιρετική έμφαση στην αξιοπιστία και τη διαθεσιμότητα.

Τόσο στο συντακτικό και τη σημασιολογία, η Erlang είναι πολύ διαφορετική από τις συμβατικές γλώσσες προγραμματισμού όπως η C ή η Java. Η Erlang χρησιμοποιεί ελαφριές "διαδικασίες" και το μήνυμα περνάει για συγχρονισμό, δεν έχει κοινόχρηστες καταστάσεις σπειρωμάτων και όλα τα δεδομένα είναι αμετάβλητα. Η στιβαρότητα, η ταυτόχρονη φύση της Erlang είναι ιδανική για ένα διακομιστή βάσης δεδομένων.

Η CouchDB έχει σχεδιαστεί για lock-free συναγωνισμό, σ’ ένα εννοιολογικό μοντέλο και στην πραγματική εφαρμογή της Erlang. Η μείωση των σημείων συμφόρησης και η αποφυγή κλειδαριών κρατά το όλο σύστημα να λειτουργεί προβλέψιμα κάτω από βαριά φορτία.

Η CouchDB μπορεί να φιλοξενήσει πολλούς πελάτες αναπαράγοντας τις αλλαγές, το άνοιγμα και την επικαιροποίηση των εγγράφων, και querying τα views των οποίων τα ευρετήρια ταυτόχρονα ανανεώνεται για άλλους πελάτες, χωρίς να χρειάζονται κλειδαριές.

Για μεγαλύτερη διαθεσιμότητα και περισσότερους χρήστες ταυτόχρονα, Η CouchDB έχει σχεδιαστεί για "shared nothing" ομαδοποίηση. Σε ένα σύμπλεγμα "shared nothing», κάθε μηχανή είναι ανεξάρτητη και αναπαράγει τα δεδομένα με τους συντρόφους του συμπλέγματος του, επιτρέποντας τις επιμέρους αποτυχίες του server με μηδενικό downtime. Και επειδή σαρώνει την συνοχή και τα fix-ups δεν είναι απαραίτητα για την επανεκκίνηση, εάν το σύνολο του συμπλέγματος αποτύχει - λόγω διακοπής ρεύματος σε ένα datacenter, για παράδειγμα - το σύνολο του κατανεμημένου συστήματος του CouchDB γίνεται άμεσα διαθέσιμο μετά την επανεκκίνηση.

Η CouchDB είναι χτισμένη από την αρχή με ένα συνεκτικό όραμα ενός κατανεμημένου συστήματος βάσεων δεδομένων documentoriented. Σε αντίθεση με επαχθείς προσπάθειες να κλειδωθεί διανέμονται χαρακτηριστικά πάνω από τα ίδια μοντέλα και τις βάσεις δεδομένων, είναι το αποτέλεσμα προσεκτικού ground-up σχεδιασμού, της μηχανικής και της ολοκλήρωσης. Τα μοντέλα των έγγραφών, του view, της ασφάλεια και της αναπαραγωγής, του ειδικού σκοπού query γλώσσας, η αποτελεσματική και ισχυρή δομή του δίσκου και η ταυτόχρονη και αξιόπιστα τη φύση της πλατφόρμας Erlang είναι προσεκτικά ενσωματωμένη για ένα αξιόπιστο και αποτελεσματικό σύστημα.

Accessing data via HTTP

Οι εφαρμογές αλληλεπιδρούν με CouchDB μέσω HTTP. Το παρακάτω δείχνει μερικά παραδείγματα χρησιμοποιώντας cURL, ένα βοηθητικό πρόγραμμα γραμμής εντολών. Αυτά τα παραδείγματα υποθέτουν ότι Η CouchDB εκτελείται σε localhost (127.0.0.1) στη θύρα 5984.

Action	Request	Response
Accessing server information	`curl http://127.0.0.1:5984/`	`{ "couchdb": "Welcome", "version":"1.1.0" }`
Creating a database named **wiki**	`curl -X PUT http://127.0.0.1:5984/wiki`	`{"ok": true}`
Attempting to create a second database named **wiki**	`curl -X PUT http://127.0.0.1:5984/wiki`	`{ "error":"file_exists", "reason":"The database could not be created, the file already exists." }`
Retrieve information about the **wiki**database	`curl http://127.0.0.1:5984/wiki`	`{ "db_name": "wiki", "doc_count": 0, "doc_del_count": 0, "update_seq": 0, "purge_seq": 0, "compact_running": false, "disk_size": 79, "instance_start_time": "1272453873691070", "disk_format_version": 5 }`
Delete the database **wiki**	`curl -X DELETE http://127.0.0.1:5984/wiki`	`{"ok": true}`

Create a document, asking CouchDB to supply a document id	`curl -X POST -H "Content-Type: application/json" --data \ '{ "text" : "Wikipedia on CouchDB", "rating": 5 }' \ http://127.0.0.1:5984/wiki`	`{ "ok": true, "id": "123BAC", "rev": "946B7D1C" }`

Η CouchDB περιλαμβάνει μια σειρά από άλλα projects ανοικτού κώδικα σαν μέρος της προεπιλεγμένης συσκευασίας του.

Component	Description	License
SpiderMonkey	SpiderMonkey is a code name for the first ever JavaScript engine, written by Brendan Eich at Netscape Communications, later released as open source and now maintained by the Mozilla Foundation.	MPL
jQuery	jQuery is a lightweight cross-browser JavaScript library that emphasizes interaction between JavaScript and HTML.	Dual license:GPL and MIT
ICU	International Components for Unicode (ICU) is an open source project of mature C/C++ and Java libraries for Unicode support, softwareinternationalization and software globalization. ICU is widely portable to many operating systems and environments.	MIT License
OpenSSL	OpenSSL is an open source implementation of the SSL and TLS protocols. The core library (written in the C programming language) implements the basic cryptographic functions and provides various utility functions.	Apache-like unique
Erlang	Erlang is a general-purpose concurrent programming language and runtime system. The sequential subset of Erlang is a functional language, with strict evaluation, single assignment, and dynamic typing.	Modified MPL

EmbeddedData

Ένα τιμολόγιο περιέχει όλες τις σχετικές πληροφορίες για μια ενιαία συναλλαγή, τον πωλητή, τον αγοραστή, την ημερομηνία, και μια λίστα με τα στοιχεία ή τις υπηρεσίες που πωλούνται. Όπως φαίνεται στο Σχήμα 1. Στα αυτοτελές έγγραφα, δεν υπάρχει αφηρημένη αναφορά σε αυτό το κομμάτι χαρτί που δείχνει ότι σε κάποιο άλλο κομμάτι χαρτί με το όνομα και τη διεύθυνση του πωλητή. Οι λογιστές εκτιμούν την απλότητα που έχει τα πάντα σε ένα μέρος. Και με δεδομένη την επιλογή, προγραμματιστές το εκτιμούν επίσης.

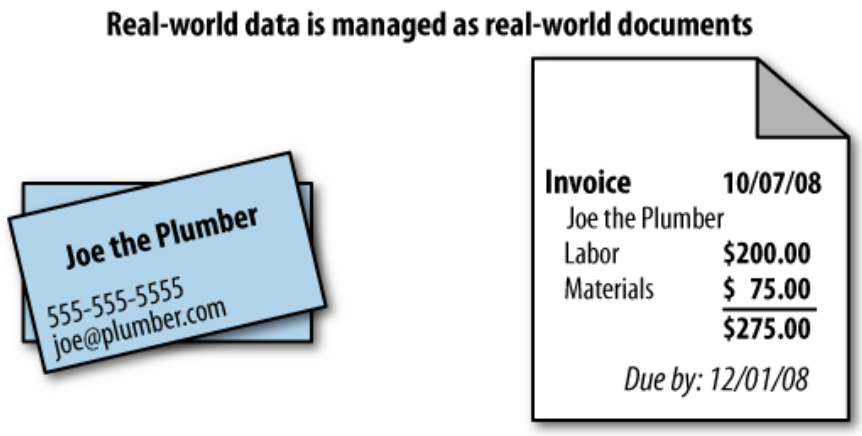


Figure 1. Self-contained documents

Ωστόσο, χρησιμοποιώντας αναφορές είναι ακριβώς το πώς θα διαμορφώσουμε τα δεδομένα μας σε μια σχεσιακή βάση δεδομένων. Κάθε τιμολόγιο αποθηκεύεται σε έναν πίνακα ως μια γραμμή που παραπέμπει σε άλλες σειρές σε άλλους πίνακες σε μία γραμμή για πληροφορίες του πωλητή, μια για τον αγοραστή, μία γραμμή για κάθε στοιχείο που τιμολογείται, και περισσότερες σειρές εξακολουθούν να περιγράφουν τις λεπτομέρειες του αντικείμενου, πληροφορίες για τον κατασκευαστή, και ούτω καθεξής.Αυτό δεν εννοείται ως δυσφήμιση του σχεσιακού μοντέλου, το οποίο είναι ευρέως εφαρμόσιμο και εξαιρετικά χρήσιμο για διάφορους λόγους. Ας ελπίσουμε, όμως, ότι δείχνει το σημείο που μερικές φορές το μοντέλο σας δεν μπορεί να «χωρέσει» τα δεδομένα σας με τον τρόπο που συμβαίνει στον πραγματικό κόσμο. Ας ρίξουμε μια ματιά στην ταπεινή βάση δεδομένων επαφών για να απεικονίζει ένα διαφορετικό τρόπο των στοιχείων μοντελοποίησης, αυτά που περισσότερο "ταιριάζουν" πραγματικού κόσμου ομόλογό του - ένα σωρό από επαγγελματικές κάρτες. Μοιάζει πολύ με το παράδειγμα του τιμολογίου μας, μια επαγγελματική κάρτα περιέχει όλες τις σημαντικές πληροφορίες, εκεί στο χαρτόνι. Καλούμε αυτό το "αυτοδύναμη" δεδομένων, και αυτό είναι μια σημαντική έννοια για την κατανόηση των βάσεων δεδομένων του εγγράφου, όπως CouchDB.

Σύνταξη και Semantics

Οι περισσότερες επαγγελματικές κάρτες περιέχουν περίπου τις ίδιες πληροφορίες - την ταυτότητα κάποιου, μια συνεργασία, και κάποια στοιχεία επικοινωνίας. Αν και η ακριβής μορφή των πληροφοριών αυτών μπορεί να διαφέρουν μεταξύ των επαγγελματικών καρτών, οι γενικές πληροφορίες που μεταφέρονται παραμένουν ίδιες, και είμαστε εύκολα σε θέση να το αναγνωρίσουμε ως επαγγελματική κάρτα. Με αυτή την έννοια, μπορούμε να περιγράψουμε μια επαγγελματική κάρτα ως ενός πραγματικού κόσμου έγγραφο.

Η επαγγελματική κάρτα του Jan θα μπορούσε να περιέχει έναν αριθμό τηλεφώνου, αλλά δεν έχει αριθμό φαξ, ενώ η επαγγελματική κάρτα του J. Chris περιέχει τόσο ένα τηλέφωνο και αριθμό φαξ. Ο Jan δεν έχει να κάνει με την έλλειψη μιας συσκευής φαξ γράφοντας κάτι τόσο γελοίο ως «Φαξ: Ουδέν» στην επαγγελματική κάρτα. Αντ 'αυτού, απλά παραλείποντας έναν αριθμό φαξ δεν σημαίνει ότι δεν έχει έναν.

Μπορούμε να δούμε ότι στον πραγματικό κόσμο τα έγγραφα του ίδιου τύπου, όπως οι επαγγελματικές κάρτες, τείνουν να είναι πολύ παρόμοια σημασιολογία και το είδος των πληροφοριών που μεταφέρουν, αλλά μπορεί να ποικίλουν σημαντικά σε σύνταξη, ή το πώς οι πληροφορίες είναι δομημένες. Ως ανθρώπινα όντα, είμαστε φυσικά άνετα που ασχολούνται με αυτό το είδος της μεταβολής.

Ενώ μια παραδοσιακή σχεσιακή βάση δεδομένων απαιτεί από εσάς να διαμορφώσει τα δεδομένα σας μπροστά, χωρίς σχηματικό σχεδιασμό σH CouchDB μπορείτε unburdens με έναν ισχυρό τρόπο για τη συγκέντρωση των δεδομένων σας μετά το γεγονός, όπως ακριβώς κάνουμε με τον πραγματικό κόσμο των εγγράφων. Θα εξετάσουμε σε βάθος το πώς να σχεδιάσουμε εφαρμογές με αυτό το υποκείμενο μοντέλο αποθήκευσης.

Χρήση σε μεγαλύτερα συστήματα

Η CouchDB είναι ένα σύστημα αποθήκευσης χρήσιμο από μόνο του. Μπορείτε να δημιουργήσετε πολλές εφαρμογές με τα εργαλεία που η CouchDBδίνει. Αλλά η CouchDB έχει σχεδιαστεί με μια ευρύτερη εικόνα στο μυαλό. Τα συστατικά του μπορούν να χρησιμοποιηθούν ως δομικά στοιχεία που επιλύουν τα προβλήματα αποθήκευσης με ελαφρώς διαφορετικούς τρόπους για μεγαλύτερες και πιο πολύπλοκα συστήματα.

Είτε θα πρέπει να έχετε ένα σύστημα που είναι τρελά γρήγορο, αλλά δεν είναι πάρα πολύ με αξιόπιστο (σκεφτείτε καταγραφής), ή ένα που εγγυάται την αποθήκευση σε δύο ή περισσότερες φυσικά διαχωρισμένες θέσεις για την αξιοπιστία του, αλλά είστε πρόθυμοι να αναλάβετε μια πτώση της απόδοσης, Η CouchDB σας επιτρέπει την κατασκευή αυτών των συστημάτων.

Υπάρχει μια πληθώρα από κουμπιά που μπορείτε να ενεργοποιήσετε για να κάνει ένα σύστημα να λειτουργεί καλύτερα σε μια περιοχή, αλλά θα επηρεάσει μια άλλη περιοχή όταν το πράξει. Ένα παράδειγμα θα ήταν το θεώρημα της CAP που συζήταει

την ενδεχόμενη Συνοχή. Για να σας δώσω μια ιδέα για άλλα πράγματα που επηρεάζουν τα συστήματα αποθήκευσης, βλέπε το σχήμα 2 και το σχήμα 3.

Με τη μείωση του χρόνου αναμονής για ένα δεδομένο σύστημα (και αυτό δεν ισχύει μόνο για τα συστήματα αποθήκευσης), επηρεάζεται ο συγχρονισμός και απόδοση ικανότητες.

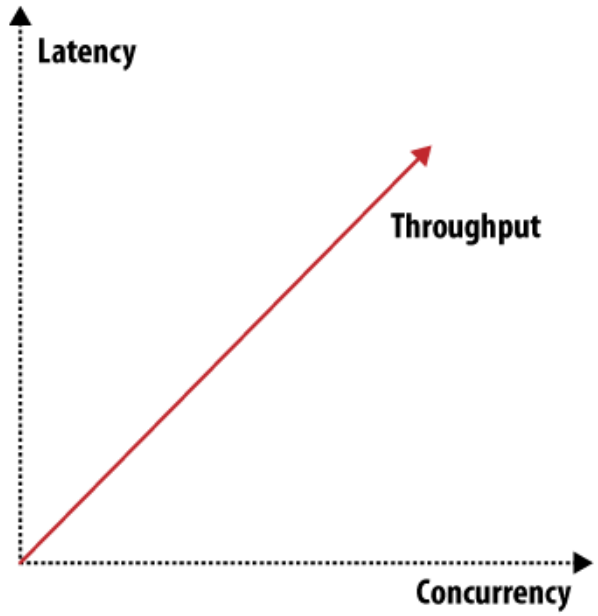


Figure 2. Throughput, latency, or concurrency

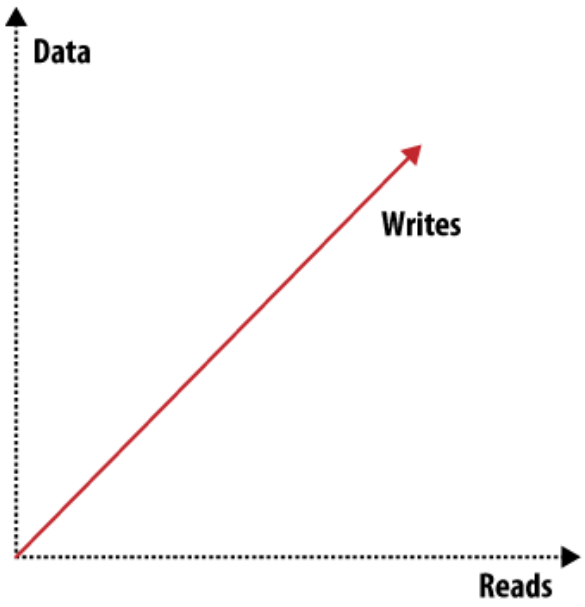


Figure 3. Scaling: read requests, write requests, or data

CouchDB Replication

Η αντιγραφή στην CouchDB είναι ένα από τα βασικά δομικά στοιχεία. Η θεμελιώδη λειτουργία του είναι να βοηθά στον συγκχρονισμό δύο ή περισσοτέρων βάσεων δεδομένων CouchDB. Αυτό μπορεί να ακούγεται απλό, αλλά η απλότητα είναι το κλειδί για να επιτρέπει την αντιγραφή για να λύσει μια σειρά από προβλήματα: για να συγχρονίσετε αξιόπιστα βάσεις δεδομένων μεταξύ πολλαπλών μηχανών για τους περιττή αποθήκευση δεδομένων όπου διανέμει τα δεδομένα σε ένα σύμπλεγμα CouchDB περιπτώσεων που μοιράζονται ένα υποσύνολο του συνολικού αριθμού των αιτήσεων που έπληξε τη συστάδα (εξισορρόπησης φορτίου) και διανομή δεδομένων μεταξύ φυσικών απομακρυσμένων τοποθεσιών, όπως ένα γραφείο στη Νέα Υόρκη και το άλλο στο Τόκιο.

Η CouchDB αντιγραφή χρησιμοποιεί το ίδια REST API καθώς όλοι οι πελάτες χρησιμοποιούν. HTTP είναι πανταχού παρούσα και καλά κατανοητή. Η αντιγραφή λειτουργεί σταδιακά? Δηλαδή, αν κατά τη διάρκεια της αντιγραφής κάτι πάει στραβά, όπως ρίψη της σύνδεσής σας στο δίκτυο, θα συνεχίσει από το σημείο που σταμάτησε την επόμενη φορά που τρέξει. Επίσης μεταφέρει δεδομένα που απαιτούνται για να συγχρονίσετε τις βάσεις δεδομένων μόνο.

Μια υπόθεση που κάνει ο πυρήνας του CouchDB είναι ότι τα πράγματα μπορούν να πάνε στραβά, όπως και τα προβλήματα σύνδεσης με το δίκτυο, και είναι σχεδιασμένο για αποκατάσταση του σφάλματος αντί να υποθέτει ότι όλα θα πάνε καλά.

Οι ιδέες πίσω από "τα πράγματα που μπορεί να πάει στραβά", οι οποίες ενσωματώνονται στις πλάνες του Distributed Computing είναι οι εξης:

- Το δίκτυο είναι αξιόπιστο.
- Το Latency είναι μηδέν.
- Το εύρος ζώνης είναι άπειρο.
- Το δίκτυο είναι ασφαλές.
- Η τοπολογία δεν αλλάζει.
- Υπάρχει ένας διαχειριστής
- Το κόστος μεταφοράς είναι μηδέν.
- Το δίκτυο είναι ομοιογενής.

Τα υπάρχοντα εργαλεία συχνά προσπαθούν να αποκρύψουν το γεγονός ότι υπάρχει ένα δίκτυο και ότι κάποια ή όλα τα προηγούμενα δεν υπάρχουν οι προϋποθέσεις για ένα συγκεκριμένο σύστημα. Αυτό συνήθως οδηγεί σε θανατηφόρα σενάρια λάθους όταν κάτι πάει στραβά τελικά. Σε αντίθεση, Η CouchDB δεν προσπαθεί να κρύψει το δίκτυο? χειρίζεται μόνο τα λάθη και με χάρη και σας επιτρέπει να γνωρίζετε όταν οι απαιτούμενες ενέργειες στο τέλος απαιτούνται.

ΚΕΦΑΛΑΙΟ 3- ΠΟΙΟΤΙΚΗ ΚΑΙ ΠΟΣΟΤΙΚΗ ΜΕΛΕΤΗ

3.1 Εισαγωγή

Στο παρόν κεφάλαιο περιγράφουμε τις ενέργειες που πραγματοποιήσαμε προκειμένου να ολοκληρώσουμε την ποιοτική και ποσοτική μελέτη. Περιγράφουμε την κατηγοριοποίηση των χαρακτηριστικών η οποία μας βοήθησε να ολοκληρώσουμε την Ποιοτική Μελέτη των 2 βάσεων καθώς και τα εργαλεία και τις ενέργειες που μας βοήθησαν να ολοκληρώσουμε την Ποσοτική σύγκριση τους.

3.2 Ποιοτική Μελέτη

Στην παρακάτω ενότητα, συγκεντρώσαμε όλα τα χαρακτηριστικά τα οποία διέπουν την λειτουργία των 2 βάσεων δεδομένων και τα χωρίσαμε σε κατηγορίες. Λεπτομέρειες μπορείτε να βρείτε παρακάτω στον συγκεντρωτικό πίνακα.

	Mongo	**Couch**
Database model	Document Store (or document based)	Document Store
Initial Release	2009	2005
System		
Server	Cross-platform	Android

Operating System	Linux OS X Solaris Windows	BSD Linux OS X Solaris Windows
Programming language	C C# C++ Erlang Haskell Java JavaScript Perl PHP Python Ruby Scala	C C# ColdFusion Erlang Haskell Java JavaScript Lisp Lua Objective-C OCaml Perl PHP PL/SQL Python Ruby Smalltalk
Written in	C++	Erlang
License	Open source: Free GNU AGPL v3.0 license	Open source: Apache 2.0 license
Value max size (or document size limit)	16MB	20MB
Design & Features http://vschart.com/compare/mongodb/vs/couchdb **[Grolinger et al. Journal of Cloud Computing: Advances, Systems and Applications 2013, 2:22]** **Moniruzzaman, Hossain, International Journal of Database Theory and Application Vol. 6, No. 4. 2013** **Sugam Sharma et al., International Journal of Big Data Intelligence 05/2015**		
Data Storage	Volatile memory, Filesystem	Volatile memory, Filesystem
Schema	No	No
Query language	Volatile memory, Filesystem/ MongoDB ad-hoc query language	Javascript , Memcached-protocol
Protocol	Custom, binary (BSON)	HTTP, REST
Conditional entry updates	Yes	Yes
MapReduce	Yes	Yes

Unicode	Yes	Yes
TTL for entries	Yes	Yes
Compression	Yes	Yes
REST	Yes	Yes
Other API and access methods	CLI and API in several languages proprietary protocol using JSON	API in several languages RESTful HTTP/JSON API
Other features	Server-side scripting and secondary indexing support. A powerful aggregation framework	Server-side scripting and secondary indexing support
Data Processing nature	Batch processing and event streaming	Batch processing
Integrity (ACID properties) **[Grolinger et al. Journal of Cloud Computing: Advances, Systems and Applications 2013, 2:22]** **Moniruzzaman, Hossain, International Journal of Database Theory and Application Vol. 6, No. 4. 2013** **Sugam Sharma et al., International Journal of Big Data Intelligence 05/2015**		
Integrity model	BASE	MVCC
Atomicity	Conditional	Yes
Consistency	Yes Configurable. 2 methods to achieve strong consistency: set connection to read only from primary, or set write concern parameter to "Replica Acknowledged"	Yes Eventual consistency
Isolation	No	Yes
Durability(data storage)	Yes	Yes
Transactions	No	No
Referential integrity	No	No
Revision Control	No	Yes
Single point of failure	No	No
Versioning	Yes (different database for audit trails, not native)	Yes (MVCC)
Concurrency Control	Readers-writer locks	MVCC. In case of conflicts, the winning revision is chosen, but the losing revision is saved as a previous version
Indexing **[Sugam Sharma et al., International Journal of Big Data Intelligence 05/2015]**		
Indexing	Advanced and wide variety (include spatial indexing)	Basic associated map and reduce functions

Secondary indexes	Yes	Yes
Composite keys	Yes	Yes
Full text search	No	No
Geospatial indexes	Yes	No
Graph Support	No	No
Distribution **[Grolinger et al. Journal of Cloud Computing: Advances, Systems and Applications 2013, 2:22]** **[Moniruzzaman, Hossain, International Journal of Database Theory and Application Vol. 6, No. 4. 2013]**		
Horizontal scalable	Yes	Yes
Replication	Yes	Yes
Replication mode	Master-Slave- Replication, asynchronous replication	Master-Slave-Replication Multi-master, asynchronous replication. Designed for off-line operation. Multiple replicas can maintain their own copies of the same data and synchronize them at a later time Master-master replication Master-slave replication
Sharding	Yes	Yes
Shared nothing architecture	Yes	Yes
Partitioning	Range partitioning based on a shard key (one or more fields that exist in every doc in the collection). In addition, hashed shard keys can be used to partition data Sharing	Consistent hashing
Security Features **[Grolinger et al. Journal of Cloud Computing: Advances, Systems and Applications 2013, 2:22]**		
Encryption		N/A
Data at rest	No, a third party partner (Gazzang) provides an encryption plugin	Yes, SSL based
Client/Server	Yes, SSL based	Possible using HTTPS connections
Server/Server	Yes	Yes, HTTP authentication using cookies or BASIC method. Oauth supported
Authentication	Yes, store credentials in a system collection. REST interface does not support authentication. Enterprise	Three levels of users: server admin, database admin and database member. Complex

	Edition supports Kerberos	authorization can be done in validation functions
Authorization	Yes, permissions include read, read/write, dbAdmin, and userAdmin. Granularity of collections.	No
Auditing	No	

3.3 Ποσοτική Μελέτη

Προκειμένου να πραγματοποιήσουμε την ποσοτική μελέτη χρησιμοποιήσαμε το YCSB, το οποίο είναι ένα BenchmarkingFrameworkγια cloudσυστήματακαι μέτρηση του performancetους. Δίνει την δυνατότητα στους developersνα δημιουργήσουν τα δικά τους Workloadsαλλά διαθέτει ήδη ένα μεγάλο φάσμα διαθέσιμων Workloadsγια testing.

Το μηχάνημα στο οποίο πραγματοποιήσαμε τα testsέχει τα ακόλουθα χαρακτηριστικά:

- **hdd**: toshiba 6gbsata3
- **cpu**: AMDFX 6300(AM3+,3,5 GHz,14 Mb)
- **ram**: 8gbgskill 1600
- **OS:** Ubuntu 12.04.2 LTS

Οι εκδόσεις των βάσεων δεδομένων που χρησιμοποιήθηκαν είναι οι εξής:

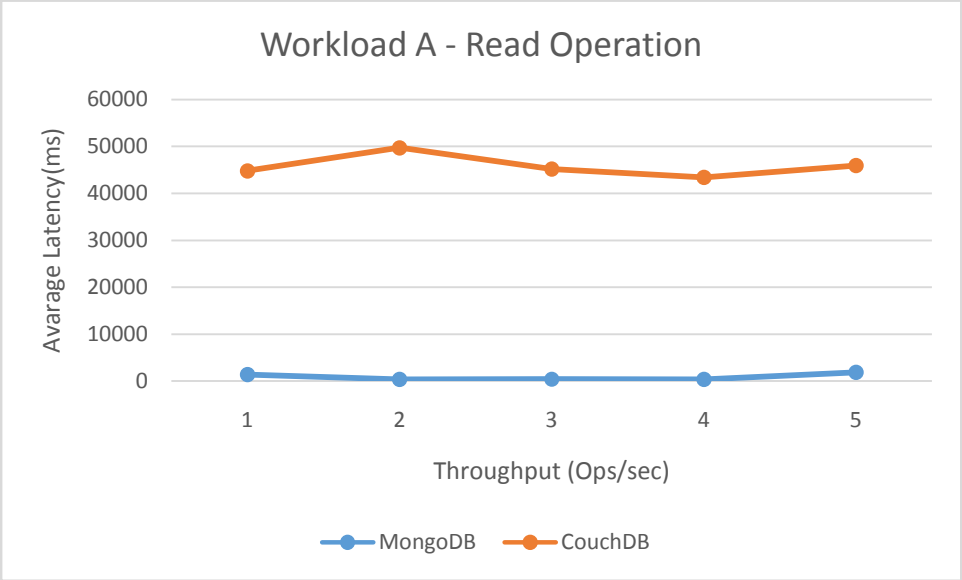
- MongoDB3.0.2
- CouchDB 1.6.1

Τα Workloadsπου χρησιμοποιήθηκαν είναι τα ακόλουθα:

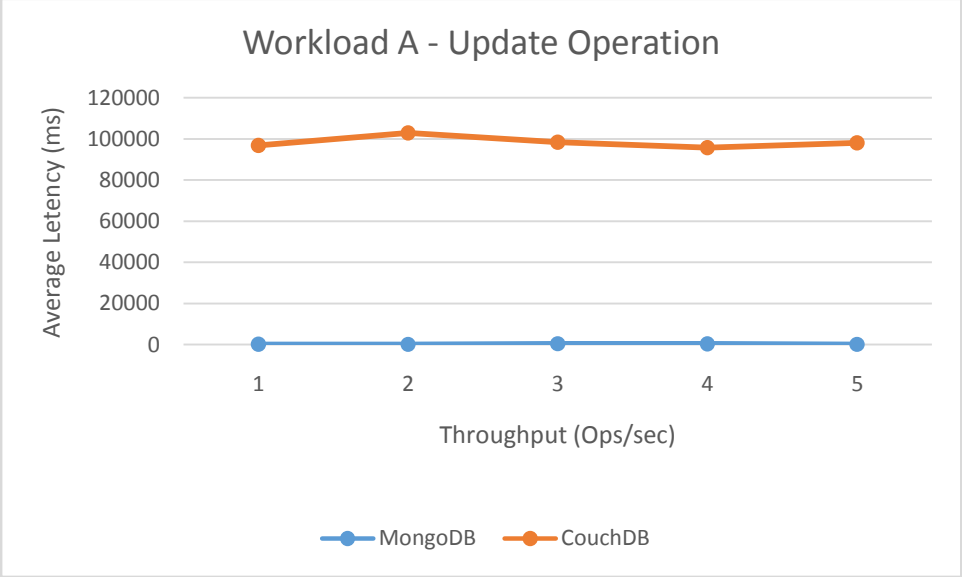
- Workload A: Update heavy workload: 50/50% read/update
- Workload B: Read mostly workload: 95/5% read/update
- Workload C: Read only: 100% reads.
- Workload D: Read latest workload: More traffic on recent inserts (95% reas, 5% insert).
- Workload E: Short ranges: Short range based queries 95-5% scan-insert
- Workload F: Read-modify-write: Read, modift and update existing records 50/50 read/read-modify-write ratio

Στα AppendicesA & B έχουμε συμπεριλάβει talogstης εκτέλεσης των Workloads.

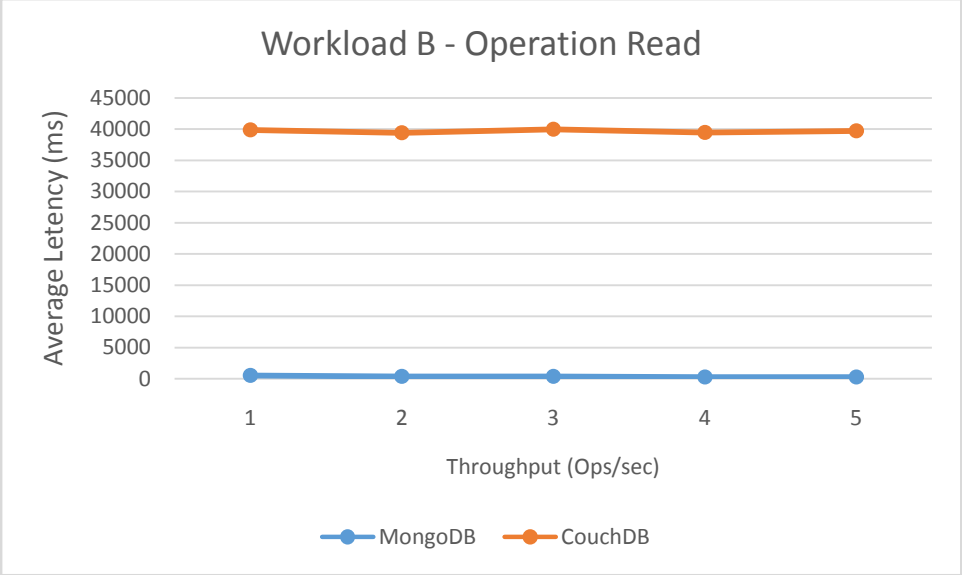
Συγκεντρωτικά τα αποτελέσματα των Testφαίνονται στα παρακάτω διαγράμματα.



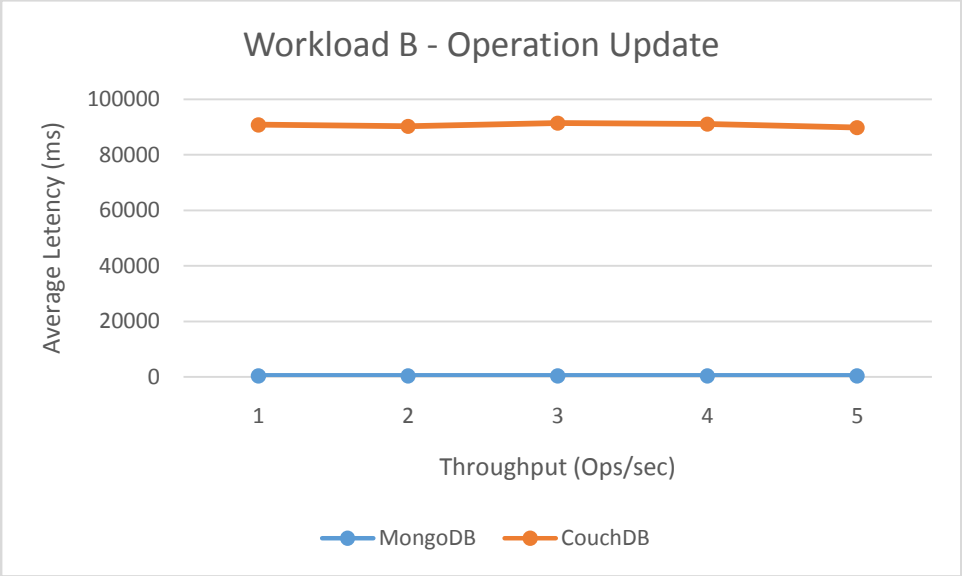
a)Workload A- Read



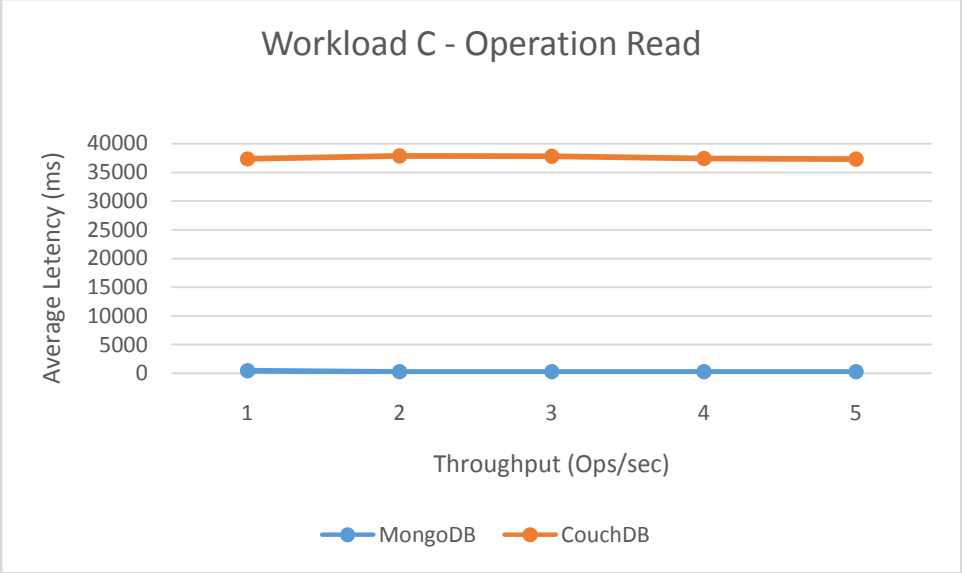
b) Workload A- Update



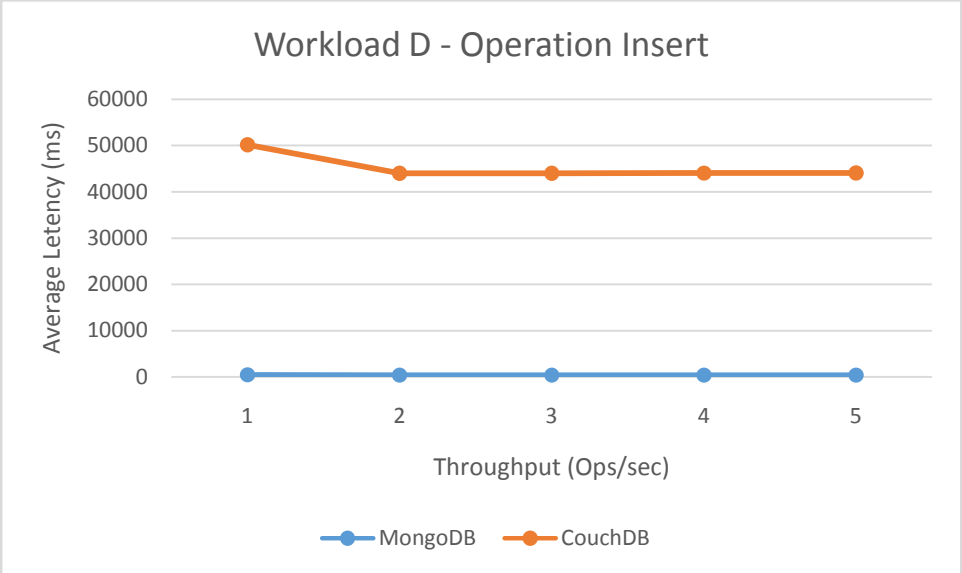
c)Workload B- Read



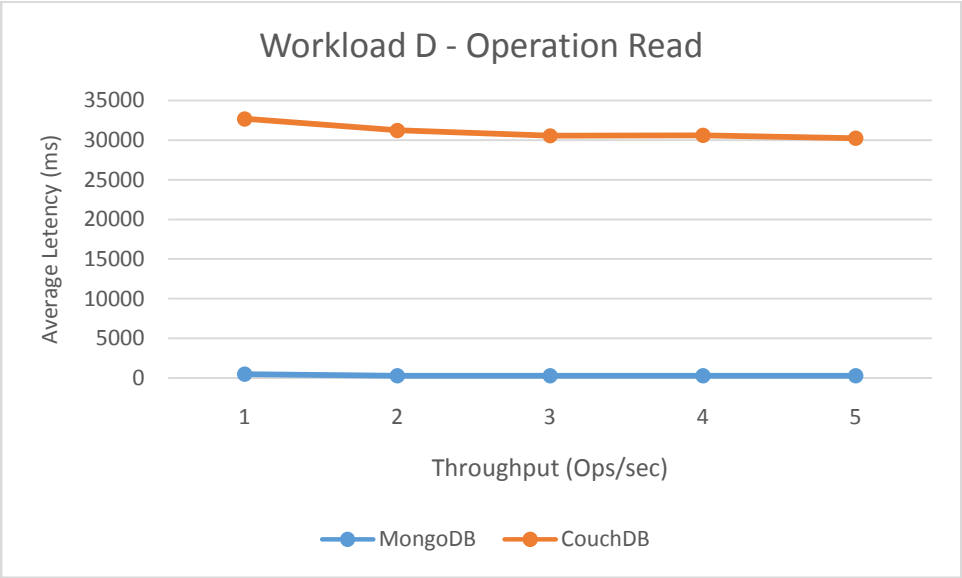
d)Workload B- Update



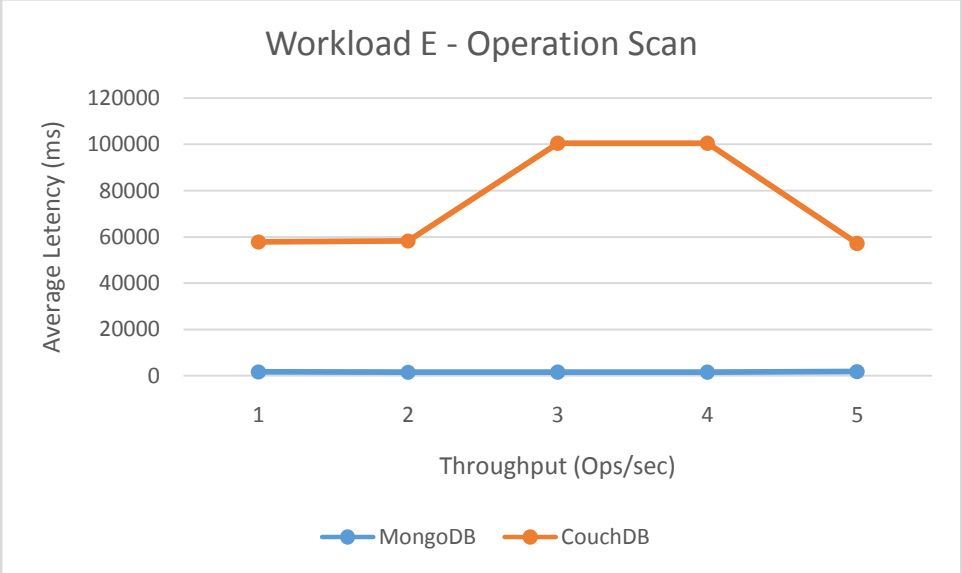
e)Workload C- Read



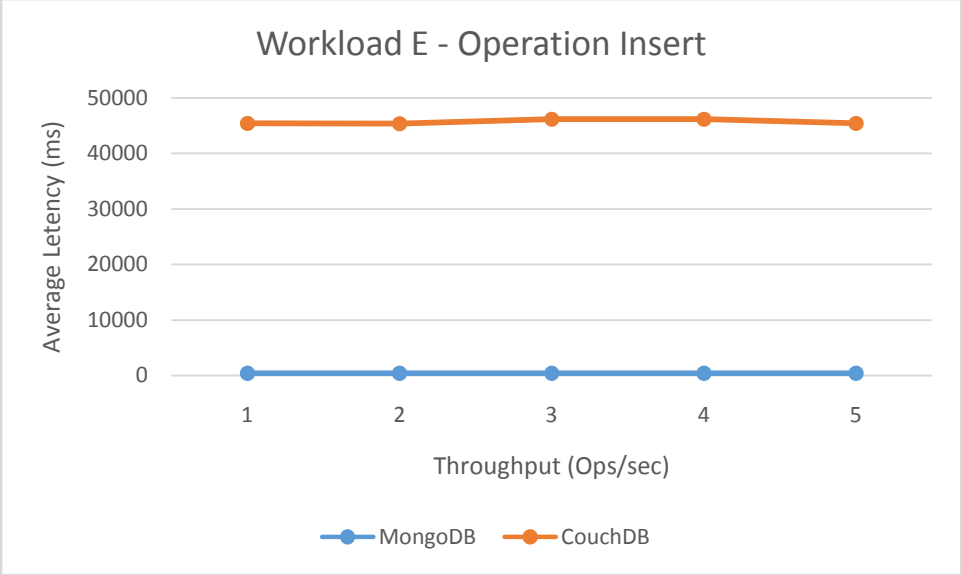
f)Workload D- Insert



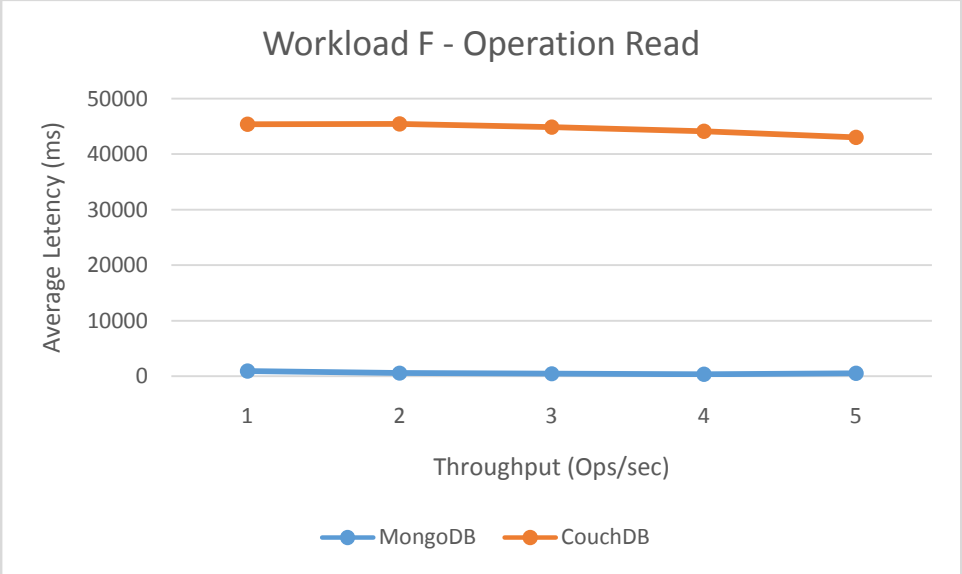
g)Workload D- Read



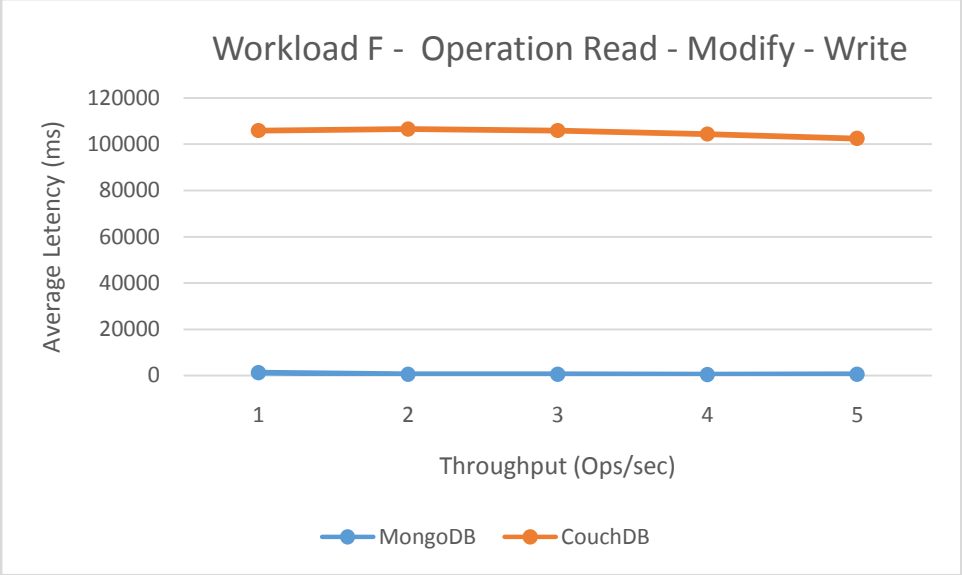
h)Workload E- Scan



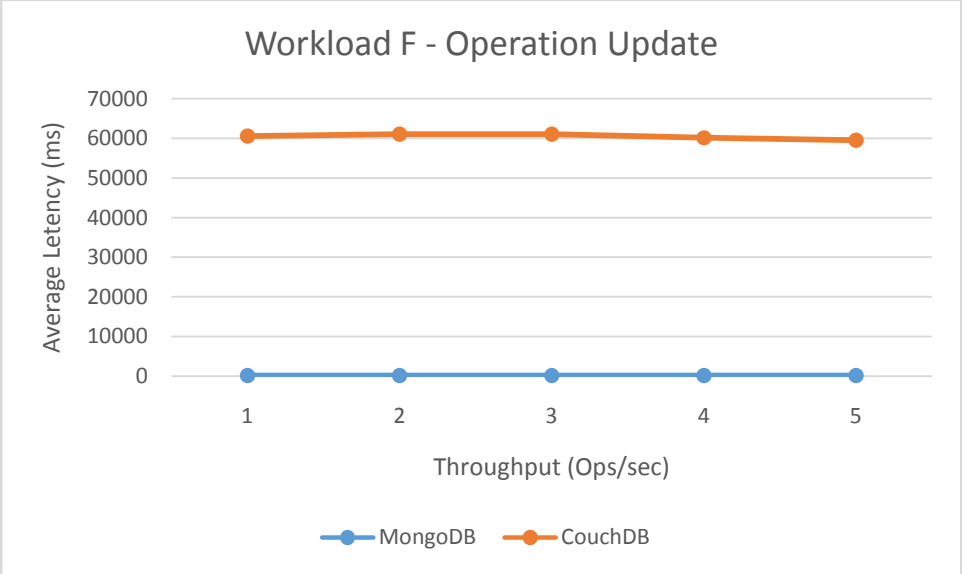
i)Workload E - Insert



J)Workload E - Read



k)Workload F – Read-Modify-Write



l)WorkloadF - Update

ΣΥΜΠΕΡΑΣΜΑΤΑ

Έπειτα από αυτή την εκτενή ποιοτική και ποσοτική έρευνα που πραγματοποιήσαμε πάνω στις NoSQL βάσεις δεδομένων, και συγκεκριμένα στην MongoDB και στην CouchDB είμαστε σε θέση να πούμε με βεβαιότητα ότι η MongoDB υπερέχει κατα πολύ της CouchDB, καθώς σε όλα τα tests τα οποία έγιναν τα αποτελέσματα είναι σαφώς καλύτερα τόσο από άποψη χρόνου, όσο throughput και latency.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] NoSQL - http://nosql-database.org/.

[2] Stonebraker, M.: SQL databases vs. NoSQL databases. Communications of the ACM, 53(4): 10-11, 2010.

[3] Dayarathna, M. and Suzumura, T.: XGDBench: A benchmarking platform for graph stores in exascale clouds. Cloud Computing Technology and Science (CloudCom), IEEE 4th International Conference on, Taipei, 363 – 370, 2012.

[4]Chang, F., Jeffrey, D., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A. and Gruber, R.: Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems, 26(2), Article 4, 2008.

[5] Apache Cassandra. http://incubator.apache.org/cassandra/.

[6] Apache CouchDB. http://couchdb.apache.org/.

AppendixA – CouchDBLogs

Load a

[OVERALL], RunTime(ms), 4.8955564E7

[OVERALL], Throughput(ops/sec), 20.426687352636772

[INSERT], Operations, 1000000

[INSERT], AverageLatency(us), 48945.823845

[INSERT], MinLatency(us), 39756

[INSERT], MaxLatency(us), 744113

[INSERT], 95thPercentileLatency(ms), 48

[INSERT], 99thPercentileLatency(ms), 183

[INSERT], Return=0, 1000000

Run a target=1000

[OVERALL], RunTime(ms), 7010781.0

[OVERALL], Throughput(ops/sec), 14.2637460790745

[READ], Operations, 50158

[READ], AverageLatency(us), 43446.99467682125

[READ], MinLatency(us), 2646

[READ], MaxLatency(us), 416081

[READ], 95thPercentileLatency(ms), 60

[READ], 99thPercentileLatency(ms), 135

[READ], Return=0, 50158

[UPDATE], Operations, 49842

[UPDATE], AverageLatency(us), 96880.10661690944

[UPDATE], MinLatency(us), 43854

[UPDATE], MaxLatency(us), 855836

[UPDATE], 95thPercentileLatency(ms), 167

[UPDATE], 99thPercentileLatency(ms), 231

[UPDATE], Return=0, 49842

Run a target=5000

[OVERALL], RunTime(ms), 7613872.0

[OVERALL], Throughput(ops/sec), 13.133921873128417

[READ], Operations, 50140

[READ], AverageLatency(us), 49422.91017151975

[READ], MinLatency(us), 2926

[READ], MaxLatency(us), 592007

[READ], 95thPercentileLatency(ms), 72

[READ], 99thPercentileLatency(ms), 151

[READ], Return=0, 50140

[UPDATE], Operations, 49860

[UPDATE], AverageLatency(us), 102947.9197954272

[UPDATE], MinLatency(us), 43806

[UPDATE], MaxLatency(us), 779960

[UPDATE], 95thPercentileLatency(ms), 175

[UPDATE], 99thPercentileLatency(ms), 231

[UPDATE], Return=0, 49860

Run a target= 10.000

[OVERALL], RunTime(ms), 7171198.0

[OVERALL], Throughput(ops/sec), 13.944671448201541

[READ], Operations, 49944

[READ], AverageLatency(us), 44853.83974050937

[READ], MinLatency(us), 2876

[READ], MaxLatency(us), 339963

[READ], 95thPercentileLatency(ms), 63

[READ], 99thPercentileLatency(ms), 123

[READ], Return=0, 49944

[UPDATE], Operations, 50056

[UPDATE], AverageLatency(us), 98464.11581029247

[UPDATE], MinLatency(us), 43897

[UPDATE], MaxLatency(us), 647849

[UPDATE], 95thPercentileLatency(ms), 167

[UPDATE], 99thPercentileLatency(ms), 220

[UPDATE], Return=0, 50056

Run a target=50.000

[OVERALL], RunTime(ms), 6947486.0

[OVERALL], Throughput(ops/sec), 14.393695791542438

[READ], Operations, 49950

[READ], AverageLatency(us), 43097.93757757758

[READ], MinLatency(us), 2779

[READ], MaxLatency(us), 228002

[READ], 95thPercentileLatency(ms), 60

[READ], 99thPercentileLatency(ms), 111

[READ], Return=0, 49950

[UPDATE], Operations, 50050

[UPDATE], AverageLatency(us), 95751.17314685315

[UPDATE], MinLatency(us), 43858

[UPDATE], MaxLatency(us), 715960

[UPDATE], 95thPercentileLatency(ms), 155

[UPDATE], 99thPercentileLatency(ms), 207

[UPDATE], Return=0, 50050

Run a target=100.000

[OVERALL], RunTime(ms), 7112859.0

[OVERALL], Throughput(ops/sec), 14.059044330837994

[READ], Operations, 49992

[READ], AverageLatency(us), 44107.55832933269

[READ], MinLatency(us), 2786

[READ], MaxLatency(us), 288071

[READ], 95thPercentileLatency(ms), 63

[READ], 99thPercentileLatency(ms), 119

[READ], Return=0, 49992

[UPDATE], Operations, 50008

[UPDATE], AverageLatency(us), 98093.69674852023

[UPDATE], MinLatency(us), 43919

[UPDATE], MaxLatency(us), 659855

[UPDATE], 95thPercentileLatency(ms), 163

[UPDATE], 99thPercentileLatency(ms), 219

[UPDATE], Return=0, 50008

Run B target =1000

[OVERALL], RunTime(ms), 4200171.0

[OVERALL], Throughput(ops/sec), 23.80855446123503

[READ], Operations, 94905

[READ], AverageLatency(us), 39358.605447552814

[READ], MinLatency(us), 2847

[READ], MaxLatency(us), 395980

[READ], 95thPercentileLatency(ms), 56

[READ], 99thPercentileLatency(ms), 103

[READ], Return=0, 94905

[UPDATE], Operations, 5095

[UPDATE], AverageLatency(us), 90796.16427870461

[UPDATE], MinLatency(us), 43862

[UPDATE], MaxLatency(us), 287157

[UPDATE], 95thPercentileLatency(ms), 127

[UPDATE], 99thPercentileLatency(ms), 211

[UPDATE], Return=0, 5095

Run b target =5000

[OVERALL], RunTime(ms), 4165439.0

[OVERALL], Throughput(ops/sec), 24.007073444119577

[READ], Operations, 95004

[READ], AverageLatency(us), 39074.86371100164

[READ], MinLatency(us), 2809

[READ], MaxLatency(us), 342478

[READ], 95thPercentileLatency(ms), 56

[READ], 99thPercentileLatency(ms), 100

[READ], Return=0, 95004

[UPDATE], Operations, 4996

[UPDATE], AverageLatency(us), 90271.403122498

[UPDATE], MinLatency(us), 46103

[UPDATE], MaxLatency(us), 256002

[UPDATE], 95thPercentileLatency(ms), 128

[UPDATE], 99thPercentileLatency(ms), 211

[UPDATE], Return=0, 4996

Run b target =10000

[OVERALL], RunTime(ms), 4223533.0

[OVERALL], Throughput(ops/sec), 23.676860107402973

[READ], Operations, 94999

[READ], AverageLatency(us), 39622.32406656912

[READ], MinLatency(us), 2725

[READ], MaxLatency(us), 423612

[READ], 95thPercentileLatency(ms), 56

[READ], 99thPercentileLatency(ms), 100

[READ], Return=0, 94999

[UPDATE], Operations, 5001

[UPDATE], AverageLatency(us), 91426.67806438712

[UPDATE], MinLatency(us), 43933

[UPDATE], MaxLatency(us), 347931

[UPDATE], 95thPercentileLatency(ms), 135

[UPDATE], 99thPercentileLatency(ms), 207

[UPDATE], Return=0, 5001

Run b target =50000

[OVERALL], RunTime(ms), 4185340.0

[OVERALL], Throughput(ops/sec), 23.892921483081423

[READ], Operations, 94917

[READ], AverageLatency(us), 39196.40473255581

[READ], MinLatency(us), 2501

[READ], MaxLatency(us), 333055

[READ], 95thPercentileLatency(ms), 56

[READ], 99thPercentileLatency(ms), 98

[READ], Return=0, 94917

[UPDATE], Operations, 5083

[UPDATE], AverageLatency(us), 91045.8990753492

[UPDATE], MinLatency(us), 43944

[UPDATE], MaxLatency(us), 807618

[UPDATE], 95thPercentileLatency(ms), 132

[UPDATE], 99thPercentileLatency(ms), 211

[UPDATE], Return=0, 5083

Run b target =100000

[OVERALL], RunTime(ms), 4200002.0

[OVERALL], Throughput(ops/sec), 23.809512471660728

[READ], Operations, 95030

[READ], AverageLatency(us), 39473.54244975271

[READ], MinLatency(us), 2755

[READ], MaxLatency(us), 347978

[READ], 95thPercentileLatency(ms), 56

[READ], 99thPercentileLatency(ms), 104

[READ], Return=0, 95030

[UPDATE], Operations, 4970

[UPDATE], AverageLatency(us), 89813.89979879276

[UPDATE], MinLatency(us), 46719

[UPDATE], MaxLatency(us), 283917

[UPDATE], 95thPercentileLatency(ms), 119

[UPDATE], 99thPercentileLatency(ms), 211

[UPDATE], Return=0, 4970

Run c target =1000

[OVERALL], RunTime(ms), 3733583.0

[OVERALL], Throughput(ops/sec), 26.783923110856247

[READ], Operations, 100000

[READ], AverageLatency(us), 37315.99458

[READ], MinLatency(us), 2843

[READ], MaxLatency(us), 423238

[READ], 95thPercentileLatency(ms), 55

[READ], 99thPercentileLatency(ms), 61

[READ], Return=0, 100000

Run c target =5000

[OVERALL], RunTime(ms), 3787028.0

[OVERALL], Throughput(ops/sec), 26.40593098334631

[READ], Operations, 100000

[READ], AverageLatency(us), 37848.86117

[READ], MinLatency(us), 2804

[READ], MaxLatency(us), 361950

[READ], 95thPercentileLatency(ms), 55

[READ], 99thPercentileLatency(ms), 63

[READ], Return=0, 100000

Run c target =10000

[OVERALL], RunTime(ms), 3779709.0

[OVERALL], Throughput(ops/sec), 26.4570632289417

[READ], Operations, 100000

[READ], AverageLatency(us), 37777.61937

[READ], MinLatency(us), 2850

[READ], MaxLatency(us), 251459

[READ], 95thPercentileLatency(ms), 55

[READ], 99thPercentileLatency(ms), 63

[READ], Return=0, 100000

Run c target =50000

[OVERALL], RunTime(ms), 3742881.0

[OVERALL], Throughput(ops/sec), 26.717386954060256

[READ], Operations, 100000

[READ], AverageLatency(us), 37408.5197

[READ], MinLatency(us), 2822

[READ], MaxLatency(us), 428036

[READ], 95thPercentileLatency(ms), 55

[READ], 99thPercentileLatency(ms), 60

[READ], Return=0, 100000

Run c target =100000

[OVERALL], RunTime(ms), 3730584.0

[OVERALL], Throughput(ops/sec), 26.805454588343274

[READ], Operations, 100000

[READ], AverageLatency(us), 37286.62998

[READ], MinLatency(us), 2630

[READ], MaxLatency(us), 215989

[READ], 95thPercentileLatency(ms), 55

[READ], 99thPercentileLatency(ms), 59

[READ], Return=0, 100000

Run d target =1000

[OVERALL], RunTime(ms), 3358868.0

[OVERALL], Throughput(ops/sec), 29.77193506860049

[INSERT], Operations, 5034

[INSERT], AverageLatency(us), 50151.85776718315

[INSERT], MinLatency(us), 40496

[INSERT], MaxLatency(us), 251236

[INSERT], 95thPercentileLatency(ms), 59

[INSERT], 99thPercentileLatency(ms), 155

[INSERT], Return=0, 5034

[READ], Operations, 94966

[READ], AverageLatency(us), 32685.350862413918

[READ], MinLatency(us), 2233

[READ], MaxLatency(us), 642620

[READ], 95thPercentileLatency(ms), 58

[READ], 99thPercentileLatency(ms), 93

[READ], Return=0, 94966

Run d target =5000

[OVERALL], RunTime(ms), 3189444.0

[OVERALL], Throughput(ops/sec), 31.353427117704527

[INSERT], Operations, 5087

[INSERT], AverageLatency(us), 44009.3137409082

[INSERT], MinLatency(us), 39908

[INSERT], MaxLatency(us), 103924

[INSERT], 95thPercentileLatency(ms), 44

[INSERT], 99thPercentileLatency(ms), 49

[INSERT], Return=0, 53

[INSERT], Return=-2, 5034

[READ], Operations, 94913

[READ], AverageLatency(us), 31223.722408942926

[READ], MinLatency(us), 2402

[READ], MaxLatency(us), 211982

[READ], 95thPercentileLatency(ms), 55

[READ], 99thPercentileLatency(ms), 63

[READ], Return=0, 94913

Run d target =10000

[OVERALL], RunTime(ms), 3122744.0

[OVERALL], Throughput(ops/sec), 32.02311812944001

[INSERT], Operations, 4953

[INSERT], AverageLatency(us), 43989.72905309913

[INSERT], MinLatency(us), 39860

[INSERT], MaxLatency(us), 81256

[INSERT], 95thPercentileLatency(ms), 44

[INSERT], 99thPercentileLatency(ms), 47

[INSERT], Return=-2, 4953

[READ], Operations, 95047

[READ], AverageLatency(us), 30541.192546845246

[READ], MinLatency(us), 2345

[READ], MaxLatency(us), 215833

[READ], 95thPercentileLatency(ms), 52

[READ], 99thPercentileLatency(ms), 63

[READ], Return=0, 95047

Run d target =50000

[OVERALL], RunTime(ms), 3131641.0

[OVERALL], Throughput(ops/sec), 31.93214036985721

[INSERT], Operations, 5135

[INSERT], AverageLatency(us), 44024.009152872444

[INSERT], MinLatency(us), 39573

[INSERT], MaxLatency(us), 107576

[INSERT], 95thPercentileLatency(ms), 44

[INSERT], 99thPercentileLatency(ms), 48

[INSERT], Return=0, 48

[INSERT], Return=-2, 5087

[READ], Operations, 94865

[READ], AverageLatency(us), 30606.604501133188

[READ], MinLatency(us), 2273

[READ], MaxLatency(us), 333173

[READ], 95thPercentileLatency(ms), 52

[READ], 99thPercentileLatency(ms), 63

[READ], Return=0, 94865

Run d target =100000

[OVERALL], RunTime(ms), 3093132.0

[OVERALL], Throughput(ops/sec), 32.32969042381638

[INSERT], Operations, 4880

[INSERT], AverageLatency(us), 44040.08606557377

[INSERT], MinLatency(us), 39954

[INSERT], MaxLatency(us), 90581

[INSERT], 95thPercentileLatency(ms), 44

[INSERT], 99thPercentileLatency(ms), 47

[INSERT], Return=-2, 4880

[READ], Operations, 95120

[READ], AverageLatency(us), 30236.386175357442

[READ], MinLatency(us), 2500

[READ], MaxLatency(us), 207938

[READ], 95thPercentileLatency(ms), 51

[READ], 99thPercentileLatency(ms), 60

[READ], Return=0, 95120

Load e

[OVERALL], RunTime(ms), 4.8198942E7

[OVERALL], Throughput(ops/sec), 20.747343375296495

[INSERT], Operations, 1000000

[INSERT], AverageLatency(us), 48189.416116

[INSERT], MinLatency(us), 39862

[INSERT], MaxLatency(us), 667955

[INSERT], 95thPercentileLatency(ms), 48

[INSERT], 99thPercentileLatency(ms), 143

[INSERT], Return=0, 1000000

Run e target =1000

[OVERALL], RunTime(ms), 5730438.0

[OVERALL], Throughput(ops/sec), 17.45067305500906

[SCAN], Operations, 94948

[SCAN], AverageLatency(us), 57819.45774529216

[SCAN], MinLatency(us), 5091

[SCAN], MaxLatency(us), 773358

[SCAN], 95thPercentileLatency(ms), 77

[SCAN], 99thPercentileLatency(ms), 121

[SCAN], Return=0, 94948

[INSERT], Operations, 5052

[INSERT], AverageLatency(us), 45415.96753760887

[INSERT], MinLatency(us), 40373

[INSERT], MaxLatency(us), 114097

[INSERT], 95thPercentileLatency(ms), 47

[INSERT], 99thPercentileLatency(ms), 49

[INSERT], Return=-2, 5052

Run f target =5000

[OVERALL], RunTime(ms), 5769247.0

[OVERALL], Throughput(ops/sec), 17.33328456902608

[SCAN], Operations, 95016

[SCAN], AverageLatency(us), 58224.18752631136

[SCAN], MinLatency(us), 4614

[SCAN], MaxLatency(us), 486683

[SCAN], 95thPercentileLatency(ms), 78

[SCAN], 99thPercentileLatency(ms), 133

[SCAN], Return=0, 95016

[INSERT], Operations, 4984

[INSERT], AverageLatency(us), 45379.874398073836

[INSERT], MinLatency(us), 40416

[INSERT], MaxLatency(us), 126609

[INSERT], 95thPercentileLatency(ms), 47

[INSERT], 99thPercentileLatency(ms), 48

[INSERT], Return=-2, 4984

Run e target =10000

[OVERALL], RunTime(ms), 9791873.0

[OVERALL], Throughput(ops/sec), 10.212550755100684

[SCAN], Operations, 94948

[SCAN], AverageLatency(us), 100555.01806251843

[SCAN], MinLatency(us), 5866

[SCAN], MaxLatency(us), 1824305

[SCAN], 95thPercentileLatency(ms), 311

[SCAN], 99thPercentileLatency(ms), 511

[SCAN], Return=0, 94948

[INSERT], Operations, 5052

[INSERT], AverageLatency(us), 46174.72585114806

[INSERT], MinLatency(us), 40262

[INSERT], MaxLatency(us), 150111

[INSERT], 95thPercentileLatency(ms), 48

[INSERT], 99thPercentileLatency(ms), 58

[INSERT], Return=0, 24

[INSERT], Return=-2, 5028

Run f target =50000

[OVERALL], RunTime(ms), 5671409.0

[OVERALL], Throughput(ops/sec), 17.632302660591044

[SCAN], Operations, 95130

[SCAN], AverageLatency(us), 57175.611962577525

[SCAN], MinLatency(us), 4557

[SCAN], MaxLatency(us), 700528

[SCAN], 95thPercentileLatency(ms), 72

[SCAN], 99thPercentileLatency(ms), 101

[SCAN], Return=0, 95130

[INSERT], Operations, 4870

[INSERT], AverageLatency(us), 45428.66755646817

[INSERT], MinLatency(us), 40263

[INSERT], MaxLatency(us), 130890

[INSERT], 95thPercentileLatency(ms), 47

[INSERT], 99thPercentileLatency(ms), 49

[INSERT], Return=-2, 4870

Run f target =100000

[OVERALL], RunTime(ms), 6009167.0

[OVERALL], Throughput(ops/sec), 16.641241623006984

[SCAN], Operations, 94972

[SCAN], AverageLatency(us), 60520.674103946425

[SCAN], MinLatency(us), 4668

[SCAN], MaxLatency(us), 870925

[SCAN], 95thPercentileLatency(ms), 91

[SCAN], 99thPercentileLatency(ms), 166

[SCAN], Return=0, 94972

[INSERT], Operations, 5028

[INSERT], AverageLatency(us), 49750.05628480509

[INSERT], MinLatency(us), 41299

[INSERT], MaxLatency(us), 214875

[INSERT], 95thPercentileLatency(ms), 51

[INSERT], 99thPercentileLatency(ms), 123

[INSERT], Return=0, 5028

Run f target =1000

[OVERALL], RunTime(ms), 7572106.0

[OVERALL], Throughput(ops/sec), 13.206365573857523

[READ], Operations, 100000

[READ], AverageLatency(us), 45400.93235

[READ], MinLatency(us), 2943

[READ], MaxLatency(us), 348071

[READ], 95thPercentileLatency(ms), 64

[READ], 99thPercentileLatency(ms), 132

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 50040

[READ-MODIFY-WRITE], AverageLatency(us), 105916.73800959233

[READ-MODIFY-WRITE], MinLatency(us), 47764

[READ-MODIFY-WRITE], MaxLatency(us), 788039

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 195

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 250

[UPDATE], Operations, 50040

[UPDATE], AverageLatency(us), 60537.691666666666

[UPDATE], MinLatency(us), 43758

[UPDATE], MaxLatency(us), 679468

[UPDATE], 95thPercentileLatency(ms), 127

[UPDATE], 99thPercentileLatency(ms), 199

[UPDATE], Return=0, 50040

Run f target =5000

[OVERALL], RunTime(ms), 7605152.0

[OVERALL], Throughput(ops/sec), 13.148981111751613

[READ], Operations, 100000

[READ], AverageLatency(us), 45450.87894

[READ], MinLatency(us), 2857

[READ], MaxLatency(us), 403972

[READ], 95thPercentileLatency(ms), 64

[READ], 99thPercentileLatency(ms), 127

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 50081

[READ-MODIFY-WRITE], AverageLatency(us), 106552.35143068229

[READ-MODIFY-WRITE], MinLatency(us), 47822

[READ-MODIFY-WRITE], MaxLatency(us), 519966

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 187

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 247

[UPDATE], Operations, 50081

[UPDATE], AverageLatency(us), 61048.43890896747

[UPDATE], MinLatency(us), 43798

[UPDATE], MaxLatency(us), 475863

[UPDATE], 95thPercentileLatency(ms), 119

[UPDATE], 99thPercentileLatency(ms), 195

[UPDATE], Return=0, 50081

Run f target =10000

[OVERALL], RunTime(ms), 7540799.0

[OVERALL], Throughput(ops/sec), 13.261194205017267

[READ], Operations, 100000

[READ], AverageLatency(us), 44866.75189

[READ], MinLatency(us), 2610

[READ], MaxLatency(us), 367972

[READ], 95thPercentileLatency(ms), 63

[READ], 99thPercentileLatency(ms), 127

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 50005

[READ-MODIFY-WRITE], AverageLatency(us), 105895.12518748126

[READ-MODIFY-WRITE], MinLatency(us), 47881

[READ-MODIFY-WRITE], MaxLatency(us), 483947

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 183

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 243

[UPDATE], Operations, 50005

[UPDATE], AverageLatency(us), 61023.81137886211

[UPDATE], MinLatency(us), 43768

[UPDATE], MaxLatency(us), 439868

[UPDATE], 95thPercentileLatency(ms), 119

[UPDATE], 99thPercentileLatency(ms), 195

[UPDATE], Return=0, 50005

Run f target =50000

[OVERALL], RunTime(ms), 7425925.0

[OVERALL], Throughput(ops/sec), 13.466335843682774

[READ], Operations, 100000

[READ], AverageLatency(us), 44141.19842

[READ], MinLatency(us), 2883

[READ], MaxLatency(us), 687968

[READ], 95thPercentileLatency(ms), 60

[READ], 99thPercentileLatency(ms), 124

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 50057

[READ-MODIFY-WRITE], AverageLatency(us), 104361.1281139501

[READ-MODIFY-WRITE], MinLatency(us), 47724

[READ-MODIFY-WRITE], MaxLatency(us), 743872

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 179

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 243

[UPDATE], Operations, 50057

[UPDATE], AverageLatency(us), 60112.52318357073

[UPDATE], MinLatency(us), 43769

[UPDATE], MaxLatency(us), 439857

[UPDATE], 95thPercentileLatency(ms), 117

[UPDATE], 99thPercentileLatency(ms), 195

[UPDATE], Return=0, 50057

Run f target =100000

[OVERALL], RunTime(ms), 7280119.0

[OVERALL], Throughput(ops/sec), 13.736039204853657

[READ], Operations, 100000

[READ], AverageLatency(us), 43021.34181

[READ], MinLatency(us), 2831

[READ], MaxLatency(us), 567916

[READ], 95thPercentileLatency(ms), 59

[READ], 99thPercentileLatency(ms), 116

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 50007

[READ-MODIFY-WRITE], AverageLatency(us), 102480.57313976044

[READ-MODIFY-WRITE], MinLatency(us), 47113

[READ-MODIFY-WRITE], MaxLatency(us), 467940

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 175

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 239

[UPDATE], Operations, 50007

[UPDATE], AverageLatency(us), 59497.56860039594

[UPDATE], MinLatency(us), 43689

[UPDATE], MaxLatency(us), 423870

[UPDATE], 95thPercentileLatency(ms), 115

[UPDATE], 99thPercentileLatency(ms), 191

[UPDATE], Return=0, 50007

Appendix B – MongoDB Logs

Load a

[OVERALL], RunTime(ms), 126742.0

[OVERALL], Throughput(ops/sec), 7890.044342049202

[INSERT], Operations, 1000000

[INSERT], AverageLatency(us), 123.877959

[INSERT], MinLatency(us), 19

[INSERT], MaxLatency(us), 28333055

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 0

[INSERT], Return=0, 1000000

Run A target=1000

[OVERALL], RunTime(ms), 108755.0

[OVERALL], Throughput(ops/sec), 919.4979541170521

[READ], Operations, 49789

[READ], AverageLatency(us), 1368.3094659462934

[READ], MinLatency(us), 63

[READ], MaxLatency(us), 5536988

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 49789

[UPDATE], Operations, 50211

[UPDATE], AverageLatency(us), 134.173826452371

[UPDATE], MinLatency(us), 17

[UPDATE], MaxLatency(us), 20498

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 50211

Run A target=5000

[OVERALL], RunTime(ms), 22840.0

[OVERALL], Throughput(ops/sec), 4378.283712784589

[READ], Operations, 50077

[READ], AverageLatency(us), 319.90089262535696

[READ], MinLatency(us), 60

[READ], MaxLatency(us), 21664

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 50077

[UPDATE], Operations, 49923

[UPDATE], AverageLatency(us), 108.73421068445406

[UPDATE], MinLatency(us), 16

[UPDATE], MaxLatency(us), 75125

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 49923

Run A target= 10.000

[OVERALL], RunTime(ms), 24666.0

[OVERALL], Throughput(ops/sec), 4054.1636260439473

[READ], Operations, 50089

[READ], AverageLatency(us), 355.3825790093633

[READ], MinLatency(us), 60

[READ], MaxLatency(us), 496909

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 50089

[UPDATE], Operations, 49911

[UPDATE], AverageLatency(us), 108.83091903588387

[UPDATE], MinLatency(us), 16

[UPDATE], MaxLatency(us), 20237

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 49911

Run a target=50.000

[OVERALL], RunTime(ms), 22741.0

[OVERALL], Throughput(ops/sec), 4397.344004221451

[READ], Operations, 50025

[READ], AverageLatency(us), 318.0835182408796

[READ], MinLatency(us), 64

[READ], MaxLatency(us), 47448

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 50025

[UPDATE], Operations, 49975

[UPDATE], AverageLatency(us), 108.48728364182091

[UPDATE], MinLatency(us), 16

[UPDATE], MaxLatency(us), 90150

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 49975

Run a target=100.000

[OVERALL], RunTime(ms), 98978.0

[OVERALL], Throughput(ops/sec), 1010.3255268847623

[READ], Operations, 50002

[READ], AverageLatency(us), 1846.9217231310747

[READ], MinLatency(us), 59

[READ], MaxLatency(us), 7250706

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 50002

[UPDATE], Operations, 49998

[UPDATE], AverageLatency(us), 105.38001520060803

[UPDATE], MinLatency(us), 16

[UPDATE], MaxLatency(us), 19986

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 49998

Run b target =1000

[OVERALL], RunTime(ms), 100298.0

[OVERALL], Throughput(ops/sec), 997.0288540150352

[READ], Operations, 94967

[READ], AverageLatency(us), 521.0251455768846

[READ], MinLatency(us), 68

[READ], MaxLatency(us), 1995902

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 94967

[UPDATE], Operations, 5033

[UPDATE], AverageLatency(us), 339.5020862308762

[UPDATE], MinLatency(us), 81

[UPDATE], MaxLatency(us), 19712

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 1

[UPDATE], Return=0, 5033

Run b target =5000

[OVERALL], RunTime(ms), 36724.0

[OVERALL], Throughput(ops/sec), 2723.0149221217735

[READ], Operations, 95012

[READ], AverageLatency(us), 355.58407359070435

[READ], MinLatency(us), 59

[READ], MaxLatency(us), 2096264

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 95012

[UPDATE], Operations, 4988

[UPDATE], AverageLatency(us), 296.0028067361668

[UPDATE], MinLatency(us), 77

[UPDATE], MaxLatency(us), 21984

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 1

[UPDATE], Return=0, 4988

Run b target =10000

[OVERALL], RunTime(ms), 37695.0

[OVERALL], Throughput(ops/sec), 2652.871733651678

[READ], Operations, 94921

[READ], AverageLatency(us), 367.0556673444233

[READ], MinLatency(us), 56

[READ], MaxLatency(us), 4877734

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 94921

[UPDATE], Operations, 5079

[UPDATE], AverageLatency(us), 289.8117739712542

[UPDATE], MinLatency(us), 60

[UPDATE], MaxLatency(us), 19979

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 1

[UPDATE], Return=0, 5079

Run b target =50000

[OVERALL], RunTime(ms), 28710.0

[OVERALL], Throughput(ops/sec), 3483.1069313827934

[READ], Operations, 95068

[READ], AverageLatency(us), 271.51190726637776

[READ], MinLatency(us), 59

[READ], MaxLatency(us), 77023

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 95068

[UPDATE], Operations, 4932

[UPDATE], AverageLatency(us), 300.8623276561233

[UPDATE], MinLatency(us), 79

[UPDATE], MaxLatency(us), 20751

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 1

[UPDATE], Return=0, 4932

Run b target =100000

[OVERALL], RunTime(ms), 29294.0

[OVERALL], Throughput(ops/sec), 3413.6683279852527

[READ], Operations, 94947

[READ], AverageLatency(us), 277.50681959408934

[READ], MinLatency(us), 59

[READ], MaxLatency(us), 77580

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 94947

[UPDATE], Operations, 5053

[UPDATE], AverageLatency(us), 302.4607164060954

[UPDATE], MinLatency(us), 63

[UPDATE], MaxLatency(us), 29165

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 1

[UPDATE], Return=0, 5053

Run c target =1000

[OVERALL], RunTime(ms), 100280.0

[OVERALL], Throughput(ops/sec), 997.207818109294

[READ], Operations, 100000

[READ], AverageLatency(us), 455.8425

[READ], MinLatency(us), 77

[READ], MaxLatency(us), 82518

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

Run c target =5000

[OVERALL], RunTime(ms), 29193.0

[OVERALL], Throughput(ops/sec), 3425.4787106498134

[READ], Operations, 100000

[READ], AverageLatency(us), 278.52492

[READ], MinLatency(us), 59

[READ], MaxLatency(us), 74525

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

Run c target =10000

[OVERALL], RunTime(ms), 27899.0

[OVERALL], Throughput(ops/sec), 3584.357862288971

[READ], Operations, 100000

[READ], AverageLatency(us), 266.02149

[READ], MinLatency(us), 58

[READ], MaxLatency(us), 72080

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

Run c target =50000

[OVERALL], RunTime(ms), 28263.0

[OVERALL], Throughput(ops/sec), 3538.1948130064043

[READ], Operations, 100000

[READ], AverageLatency(us), 269.61975

[READ], MinLatency(us), 60

[READ], MaxLatency(us), 81192

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

Run c target =100000

[OVERALL], RunTime(ms), 28184.0

[OVERALL], Throughput(ops/sec), 3548.112404200965

[READ], Operations, 100000

[READ], AverageLatency(us), 268.95985

[READ], MinLatency(us), 57

[READ], MaxLatency(us), 70741

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

Run d target =1000

[OVERALL], RunTime(ms), 100280.0

[OVERALL], Throughput(ops/sec), 997.207818109294

[INSERT], Operations, 5012

[INSERT], AverageLatency(us), 398.83040702314446

[INSERT], MinLatency(us), 91

[INSERT], MaxLatency(us), 22847

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 5012

[READ], Operations, 94988

[READ], AverageLatency(us), 462.7951320166758

[READ], MinLatency(us), 77

[READ], MaxLatency(us), 98012

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 94988

Run d target =5000

[OVERALL], RunTime(ms), 29285.0

[OVERALL], Throughput(ops/sec), 3414.717432132491

[INSERT], Operations, 5014

[INSERT], AverageLatency(us), 359.5656162744316

[INSERT], MinLatency(us), 78

[INSERT], MaxLatency(us), 22399

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 5014

[READ], Operations, 94986

[READ], AverageLatency(us), 275.12122839155245

[READ], MinLatency(us), 58

[READ], MaxLatency(us), 71027

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 94986

Run d target =10000

[OVERALL], RunTime(ms), 29175.0

[OVERALL], Throughput(ops/sec), 3427.592116538132

[INSERT], Operations, 4940

[INSERT], AverageLatency(us), 357.8748987854251

[INSERT], MinLatency(us), 80

[INSERT], MaxLatency(us), 22076

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 4940

[READ], Operations, 95060

[READ], AverageLatency(us), 274.1203871239217

[READ], MinLatency(us), 58

[READ], MaxLatency(us), 311230

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 95060

Run d target =50000

[OVERALL], RunTime(ms), 28932.0

[OVERALL], Throughput(ops/sec), 3456.380478363058

[INSERT], Operations, 5045

[INSERT], AverageLatency(us), 350.6830525272547

[INSERT], MinLatency(us), 83

[INSERT], MaxLatency(us), 22686

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 5045

[READ], Operations, 94955

[READ], AverageLatency(us), 271.9890369122216

[READ], MinLatency(us), 58

[READ], MaxLatency(us), 78343

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 94955

Run d target =100000

[OVERALL], RunTime(ms), 28796.0

[OVERALL], Throughput(ops/sec), 3472.704542297541

[INSERT], Operations, 5038

[INSERT], AverageLatency(us), 352.27153632393805

[INSERT], MinLatency(us), 76

[INSERT], MaxLatency(us), 24247

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 5038

[READ], Operations, 94962

[READ], AverageLatency(us), 270.6646237442345

[READ], MinLatency(us), 57

[READ], MaxLatency(us), 132434

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 94962

Load e

[OVERALL], RunTime(ms), 126602.0

[OVERALL], Throughput(ops/sec), 7898.769371731884

[INSERT], Operations, 1000000

[INSERT], AverageLatency(us), 123.912047

[INSERT], MinLatency(us), 19

[INSERT], MaxLatency(us), 27097482

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 0

[INSERT], Return=0, 1000000

Run e target =1000

[OVERALL], RunTime(ms), 150441.0

[OVERALL], Throughput(ops/sec), 664.7124121748725

[SCAN], Operations, 94931

[SCAN], AverageLatency(us), 1546.410772034425

[SCAN], MinLatency(us), 135

[SCAN], MaxLatency(us), 136895

[SCAN], 95thPercentileLatency(ms), 2

[SCAN], 99thPercentileLatency(ms), 3

[SCAN], Return=0, 94931

[INSERT], Operations, 5069

[INSERT], AverageLatency(us), 396.7074373643717

[INSERT], MinLatency(us), 100

[INSERT], MaxLatency(us), 20195

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 5069

Run f target =5000

[OVERALL], RunTime(ms), 143696.0

[OVERALL], Throughput(ops/sec), 695.9135953679992

[SCAN], Operations, 94903

[SCAN], AverageLatency(us), 1475.8666111714067

[SCAN], MinLatency(us), 130

[SCAN], MaxLatency(us), 106298

[SCAN], 95thPercentileLatency(ms), 2

[SCAN], 99thPercentileLatency(ms), 3

[SCAN], Return=0, 94903

[INSERT], Operations, 5097

[INSERT], AverageLatency(us), 398.64292721208557

[INSERT], MinLatency(us), 105

[INSERT], MaxLatency(us), 19222

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 5097

Run f target =10000

[OVERALL], RunTime(ms), 147857.0

[OVERALL], Throughput(ops/sec), 676.3291558735806

[SCAN], Operations, 95059

[SCAN], AverageLatency(us), 1517.1409545650597

[SCAN], MinLatency(us), 120

[SCAN], MaxLatency(us), 278494

[SCAN], 95thPercentileLatency(ms), 2

[SCAN], 99thPercentileLatency(ms), 3

[SCAN], Return=0, 95059

[INSERT], Operations, 4941

[INSERT], AverageLatency(us), 409.98846387370975

[INSERT], MinLatency(us), 105

[INSERT], MaxLatency(us), 21407

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 4941

Run f target =50000

[OVERALL], RunTime(ms), 148479.0

[OVERALL], Throughput(ops/sec), 673.4959152472741

[SCAN], Operations, 94931

[SCAN], AverageLatency(us), 1525.695389282742

[SCAN], MinLatency(us), 106

[SCAN], MaxLatency(us), 133532

[SCAN], 95thPercentileLatency(ms), 2

[SCAN], 99thPercentileLatency(ms), 3

[SCAN], Return=0, 94931

[INSERT], Operations, 5069

[INSERT], AverageLatency(us), 400.1207338725587

[INSERT], MinLatency(us), 99

[INSERT], MaxLatency(us), 23549

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 5069

Run f target =100000

[OVERALL], RunTime(ms), 164495.0

[OVERALL], Throughput(ops/sec), 607.921213410742

[SCAN], Operations, 95035

[SCAN], AverageLatency(us), 1691.8941653075183

[SCAN], MinLatency(us), 122

[SCAN], MaxLatency(us), 362911

[SCAN], 95thPercentileLatency(ms), 2

[SCAN], 99thPercentileLatency(ms), 4

[SCAN], Return=0, 95035

[INSERT], Operations, 4965

[INSERT], AverageLatency(us), 408.4849949647533

[INSERT], MinLatency(us), 101

[INSERT], MaxLatency(us), 25278

[INSERT], 95thPercentileLatency(ms), 0

[INSERT], 99thPercentileLatency(ms), 1

[INSERT], Return=0, 4965

Run f target =1000

[OVERALL], RunTime(ms), 112489.0

[OVERALL], Throughput(ops/sec), 888.9758109681835

[READ], Operations, 100000

[READ], AverageLatency(us), 917.89494

[READ], MinLatency(us), 62

[READ], MaxLatency(us), 16387202

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 49812

[READ-MODIFY-WRITE], AverageLatency(us), 1258.9227696137477

[READ-MODIFY-WRITE], MinLatency(us), 88

[READ-MODIFY-WRITE], MaxLatency(us), 16388078

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 1

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 2

[UPDATE], Operations, 49812

[UPDATE], AverageLatency(us), 115.40146952541556

[UPDATE], MinLatency(us), 19

[UPDATE], MaxLatency(us), 20383

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 49812

Run f target =5000

[OVERALL], RunTime(ms), 61277.0

[OVERALL], Throughput(ops/sec), 1631.9336782153173

[READ], Operations, 100000

[READ], AverageLatency(us), 540.60853

[READ], MinLatency(us), 62

[READ], MaxLatency(us), 4725101

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 49927

[READ-MODIFY-WRITE], AverageLatency(us), 534.4255012317984

[READ-MODIFY-WRITE], MinLatency(us), 84

[READ-MODIFY-WRITE], MaxLatency(us), 2970529

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 1

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 2

[UPDATE], Operations, 49927

[UPDATE], AverageLatency(us), 111.20449856790914

[UPDATE], MinLatency(us), 19

[UPDATE], MaxLatency(us), 20276

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 49927

Run f target =10000

[OVERALL], RunTime(ms), 49232.0

[OVERALL], Throughput(ops/sec), 2031.1992200194995

[READ], Operations, 100000

[READ], AverageLatency(us), 420.6539

[READ], MinLatency(us), 61

[READ], MaxLatency(us), 1849247

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 50132

[READ-MODIFY-WRITE], AverageLatency(us), 551.3128740126067

[READ-MODIFY-WRITE], MinLatency(us), 83

[READ-MODIFY-WRITE], MaxLatency(us), 1849411

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 0

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 2

[UPDATE], Operations, 50132

[UPDATE], AverageLatency(us), 109.56925716109471

[UPDATE], MinLatency(us), 19

[UPDATE], MaxLatency(us), 22517

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 50132

Run f target =50000

[OVERALL], RunTime(ms), 38413.0

[OVERALL], Throughput(ops/sec), 2603.2853461067866

[READ], Operations, 100000

[READ], AverageLatency(us), 316.15802

[READ], MinLatency(us), 60

[READ], MaxLatency(us), 2228669

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 49857

[READ-MODIFY-WRITE], AverageLatency(us), 397.3105882824879

[READ-MODIFY-WRITE], MinLatency(us), 81

[READ-MODIFY-WRITE], MaxLatency(us), 459649

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 0

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 2

[UPDATE], Operations, 49857

[UPDATE], AverageLatency(us), 103.89696532081754

[UPDATE], MinLatency(us), 18

[UPDATE], MaxLatency(us), 20122

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 49857

Run f target =100000

[OVERALL], RunTime(ms), 56181.0

[OVERALL], Throughput(ops/sec), 1779.9611968459087

[READ], Operations, 100000

[READ], AverageLatency(us), 492.02511

[READ], MinLatency(us), 59

[READ], MaxLatency(us), 2339555

[READ], 95thPercentileLatency(ms), 0

[READ], 99thPercentileLatency(ms), 1

[READ], Return=0, 100000

[READ-MODIFY-WRITE], Operations, 49883

[READ-MODIFY-WRITE], AverageLatency(us), 503.50099232203354

[READ-MODIFY-WRITE], MinLatency(us), 81

[READ-MODIFY-WRITE], MaxLatency(us), 1660979

[READ-MODIFY-WRITE], 95thPercentileLatency(ms), 0

[READ-MODIFY-WRITE], 99thPercentileLatency(ms), 2

[UPDATE], Operations, 49883

[UPDATE], AverageLatency(us), 106.83136539502436

[UPDATE], MinLatency(us), 18

[UPDATE], MaxLatency(us), 20141

[UPDATE], 95thPercentileLatency(ms), 0

[UPDATE], 99thPercentileLatency(ms), 0

[UPDATE], Return=0, 49883