Μη Σχεσιακές Βάσεις Δεδομένων : Redis

Ευαγγέλου Σωτήριος

Προχωρημένη Διαχείριση Δεδομένων 2018-19
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας, Βόλος
sevangelou@e-ce.uth.gr

Περίληψη Στο πλαίσιο του μαθήματος της Προχωρημένης Διαχείρισης Δεδομένων HY428, ασχολήθηκα με το Redis, μια πλατφόρμα που χρησιμοποιείται κυρίως σαν μη σχεσιακή Βάση Δεδομένων της κατηγορίας key-value store και χαρακτηρίζεται από την ταχύτητα και την απόδοση της. Συγκεκριμένα, στο παρών report θα αναλυθούν με μορφή ενοτήτων η αρχιτεκτονική της βάσης, ο τρόπος που ανταποκρίνεται σε Μεγάλα Δεδομένα, η αποδοχή και υιοθέτηση της στην παραγωγή, οι μορφές εφαρμογών για τις οποίες είναι κατάλληλη, οι τρόποι υλοποίησης τεχνικών sharding, replication,availability κ.α., τα πλεονεκτήματα της συγκριτικά με σχεσιακές βάσεις, καθώς και διάφορες πηγές που μπορεί να χρησιμοποιήσει ο αναγνώστης για να εξικοιωθεί και να χρησιμοποιήσει αποτελεσματικά τη βάση.

1 Αρχιτεκτονική και Τεχνικά Χαρακτηριστικά

Σύμφωνα με την περιγραφή του Redis στην ιστοσελίδα του, πρόκειται για μια ανοιχτού κώδικα in-memory store δομών δεδομένων, που χρησιμοποιείται ως βάση δεδομένων, μνήμη cache ή message broker (προγράμματα-ενδιάμεσοι που μεταφράζουν μηνύματα από ένα φορμαλιστικό πρωτόκολλο μηνυμάτων σε άλλο). Όσον αφορά την αρχιτεκτονική της πλατφόρμας μας ένδιαφέρει η λέξη in-memory, που περιγράφει το σκεπτικό κατασκευής της.

1.1 Γλώσσα Προγραμματισμού

Το Redis έχει εξ ολοκλήρου αναπτυχθεί στην ”χαμηλότερη” γλώσσα υψηλού επιπέδου, την **C**. Συγκεκριμένα ακολουθεί το πρότυπο σύστημα ANSI. Ο προγραμματισμός σε C εγγυάται την γρήγορη εκτέλεση κώδικα, μέσω της άμεσης μετάφρασης της σε κώδικα μηχανής, όπως εγγυάται και την αποτελεσματική διάχειριση της μνήμης καθώς δίνει την δυνατότητα στον προγραμματιστή να διαχειρίζεται χειροκίνητα την μνήμη αν το επιθυμεί. Η απόδοση του Redis από πλευράς ταχύτητας και διαχείρισης μνήμης το κάνει ελκυστικό για χρήση στην παραγωγή.

1.2 In-memory Βάση

Όμως, η ταχύτητα του Redis δεν βασίζεται μόνο στην γλώσσα στην οποία γράφτηκε. Το Redis είναι μια **in-memory** βάση δεδομένων. Αυτό σημαίνει πώς αποθηκεύει τα δεδομένα του σε συσκευές πτητικής μνήμης, δηλαδή μνήμης που χάνεται με την διακοπή της ρευματοδότησης. Αποθηκεύει δηλαδή τα δεδομένα στην κύρια μνήμη (RAM) και όχι σε κάποιο σκληρό δίσκο, και βασίζει την ταχύτητα του στην μεγάλη ταχύτητα εγγραφής και ανάγνωσης που παρέχεται από αυτές τις συσκευές.
Φυσικά, εδώ έχουμε ένα σοβαρό **trade-off**. Το dataset που επεξεργαζόμαστε κάθε στιγμή δεν μπορεί να είναι μεγαλύτερο σε μέγεθος από την κύρια μνήμη. Για τον παραπάνω λόγο το Redis προσφέρει διάφορες τεχνικές backup αλλά και partitioning που θα αναλυθούν στην συνέχεια του report.
Ένας ακόμη λόγος που προτιμήθηκε η χρήση της κύριας μνήμης από την ομάδα που ανέπτυξε το Redis, είναι γιατί η αναπαράσταση στην κύρια μνήμη δομών δεδομένων (βλ. data structure store) είναι πιο απλή και μεταχειρίσιμη συγκριτικά με την μνήμη του δίσκου.

1.3 Παράλληλα ή Ταυτόχρονα;

Το Redis είναι κατά κύριο λόγο single threaded δηλαδή τρέχει όλο το κύριο ”σώμα δουλειάς” σε ένα μοναδικό νήμα. Κάποιος θα μπορούσε να αναρωτηθεί, γιατί να μην χρησιμοποιηθούν παραπάνω νήματα ώστε να επιταχυνθεί η διαδικασία;
Η απάντηση στο ερώτημα αυτό είναι ένας συνδυασμός από λόγους.
Αρχικά, το Redis σπάνια χρειάζεται τόση υπολογιστική δύναμη ώστε να χρειαστεί παραπάνω νήματα. Συνήθως είναι bound στο δίκτυο ή την μνήμη (βλ. Μικρά datasets στην RAM). Ένα instance του Redis σε ένα νήμα έχει την δυνατότητα να διαχειρίζεται μέχρι

2^{32} κλειδιά, οπότε η μνήμη γίνεται συνήθως το bottleneck της απόδοσης του.
Από την άλλη, το Redis επωφελείται από την χρήση ταυτόχρονου προγραμματισμού. Λειτουργεί με forks και εκτελεί τις διάφορες εργασίες σε μορφή ξεχωριστών διεργασιών. Πολλά στοιχεία δικαιολογούν αυτή την επιλογή όπως το non-blocking I/O που παρέχει, ο προγραμματισμός με τρόπο τέτοιο ώστε κάθε εργασία να λειτουργεί απρόσκοπτα και να μην υπάρχει η καθυστέρηση λόγω των εναλλαγών των εργασιών, αλλά και η ατομικότητα των εντολών που παρέχει στους χρήστες, πράγμα που επιτρέπει την μη χρήση μεθόδων συγχρονισμού (πχ.σηματοφόρων ή locks) χωρίς να τίθεται σε κίνδυνο από race conditions η λειτουργία της πλατφόρμας.

2 Redis και Big Data

Το Redis, όπως αναφέρθηκε και προηγουμένως πάσχει από αδυναμία του να χειριστεί datasets που ξεπερνούν σε μέγεθος το μέγεθος της RAM. Κάποιος θα έλεγε ότι για αυτό δεν έχει χρησιμότητα στα λεγόμενα Big Data, όμως αυτό δεν ισχύει.
Τα Big Data ή σε απλά ελληνικά τα Μεγάλα Δεδομένα αφορούν την τεράστια μάζα δεδομένων που παράγεται καθημερινά στη σύγχρονη εποχή, και τα οποία μπορούμε να χειριστούμε ώστε να επιτύχουμε συγκεκριμένους σκοπούς, όπως sentiment analysis, weather forecasting και άλλα. Ο χαρακτηρισμός μεγάλα αναφέρεται τόσο στο μέγεθος όσο και στην μεγάλη ταχύτητα με την οποία παράγονται και συνεπώς πρέπει και να δέχονται επεξεργασία.
Το Redis παρέχει στους χρήστες του τεχνικές για διαχείριση Big Data.
Αρχικά, στις δομές δεδομένων που υποστηρίζονται, περιλαμβάνονται και Streams δεδομένων, δηλαδή ροές από δεδομένα που πιθανώς παράγονται σε πραγματικό χρόνο και περνάνε διαδοχικά στη βάση μας για να περάσουν στη συνέχεια στην επεξεργασία. Φυσικά, αυτό απαντάει στο πρόβλημα της ταχύτητας των δεδομένων αλλά όχι στο μείζων πρόβλημα του μεγέθους. Ας πάρουμε ένα παράδειγμα δεδομένων μεγέθους 50TB. Για να τα χειριστεί επιτυχώς το Redis θα χρειαζόταν ο χρήστης έναν υπολογιστή με RAM μεγέθους 50TB. Αυτό φαντάζει λίγο αδύνατο, όμως ένα cluster υπολογιστών με συνολικό μέγεθος RAM 50 TB αθροιστικά δεν είναι και τόσο αδύνατο να επιτευχθεί. Το Redis παρέχει δυνατότητα partitioning που θα αναλυθεί σε βάθος στην ενότητα των ενσωματωμένων τεχνικών του. Με το partitioning ο χρήστης έχει τη δυνατότητα να μοιράσει τα δεδομένα του με τρόπο που επιθυμεί σε παραπάνω από έναν υπολογιστές και να τα χειρίζεται επιτυχώς.
Στη σύγχρονη εποχή όποιο άτομο ή οργανισμός χειρίζεται Big Data, κάνει χρήση υποδομών cloud για να επιτύχει το σκοπό του. Αυτή τη δυνατότητα του λεγόμενου scaling out παρέχει και το Redis, επιτρέπωντας τον διαμοιρασμό των δεδομένων σε πολλαπλά υπολογιστικά συστήματα, με υποστηριζόμενο μέγεθος μνήμης το άθροισμα των επιμέρους RAM τους. Πέρα από την άυξηση της μνήμης το cloud οδηγεί σε καλύτερη λειτουργία του δικτύου αλλά και πιο μεγάλη υπολογιστική δύναμη για τις λειτουργίες των BigData εφαρμογών που χρησιμοποιούν το Redis στο backend τους.

3 Το Redis στην παραγωγή

Το Redis είναι ευρέως γνωστό και χρησιμοποιούμενο στην παραγωγή από εταιρίες μεγάλου βελινεκούς. Η κύρια χρήση του είναι σαν cache store, ενώ η χρήση του σαν βάση είναι λίγο πιο σπάνια σε μεγάλες εταιρίες.
Συγκεκριμένα μεγάλες εταιρίες-εφαρμογές που χρησιμοποιούν το Redis περιλαμβάνουν:

- Twitter: Πλατφόρμα Social Media εστιασμένο σε σύντομα posts κειμένου.
- Instagram: Πλατφόρμα Social Media εστιασμένο στις φωτογραφίες.
- Tumblr: Πλατφόρμα δημιουργίας και διαχείρισης Blogspots.
- Twitch: Το μεγαλύτερο live streaming site.
- StackOverflow: Γνωστό site-forum με ερωτοαπαντήσεις προγραμματισμού.
- Github: Η μεγαλύτερη πλατφόρμα προγραμματισμού και χρήσης git.
- Coinbase: Το πιο διάσημο Bitcoin wallet.
- Medium: Πλατφόρμα δημοσιοποίησης άρθρων και blogs.
- Bleacher Report: Μεγάλο αθλητικό δημοσιογραφικό site.

Αρκετές ακόμη εταιρίες χρησιμοποιούν το redis στην παραγωγή τους.

4 Εφαρμογές

Το Redis όπως έχει ήδη ειπωθεί είναι γνωστό για την μεγάλη ταχύτητα που προσφέρει στους χρήστες του. Για τον παραπάνω λόγο χρησιμοποιείται κατά κύριο λόγο σε εφαρμογές που απαιτούν μεγάλη ταχύτητα όπως τις μνήμες cache για την επιτάχυνση εφαρμογών web.
Συγκεκρίμένα, και επαναλαμβάνοντας τα περιεχόμενα του άρθρου ”Top 5 redis use cases”, οι κύριες πέντε εφαρμογές του Redis είναι:

4.1 Session Cache

Η μεγάλη ταχύτητα του Redis το καθιστά ανάμεσα στις πρώτες επιλογές για cache εφαρμογές. Συγκεκριμένα αναλαμβάνει να αποτελέσει την cache ενός session για παράδειγμα ενός χρήστη σε μια web εφαρμογή εμπορίου. Το persistence που παρέχει του δίνει πλεονέκτημα απέναντι σε άλλα session stores όπως το Memcached, καθώς κανένας πελάτης δεν θα ήθελε για παράδειγμα να χάσει τα προϊόντα που έχει τοποθετήσει στο shopping cart του κατά τη διάρκεια του session του σε ένα site αγορών. Ακόμη, η δυνατότητα του Redis να αποθηκεύει πολλών τύπων δομές δεδομένων δίνει ακόμα ένα λόγο στην επιλογή του για τέτοιου τύπου εφαρμογές.

4.2 Full Page Cache (FPC)

Το Redis παρέχει την δυνατότητα και για δημιουργία Full Page Cache τεχνικών κατά των οποίων αποθηκεύονται σε μορφή cache ολόκληρες ιστοσελίδες, ώστε να προσφέρεται ταχύτατο page loading στους καταναλωτές του εκάστοτε κατόχου της εφαρμογής. Μέσω του persistence του Redis, ακόμα και κατά την επανεκκίνηση των redis instances οι χρήστες δεν θα παρατηρήσουν μείωση στην ταχύτητα με την οποία φορτώνονται οι σελίδες τους.

4.3 Message Queue

Η in-memory φύση του Redis που πραγματοποιεί list and set λειτουργίες στην κύρια μνήμη το καθιστά μια καταπληκτική πλατφόρμα για message queues. Πραγματοποιεί pushes και pops από γραμμικές λίστες και άλλων τύπων δομές δεδομένων με πολύ γρήγορο και απλό τρόπο και για αυτόν τον λόγο είναι αρκετά καλή επιλογή σαν message queue ή message broker, για παράδειγμα για την περίπτωση που έχουμε πληθώρα ατόμων που θέλει να πάρει και να δώσει τιμές σε μια δομή δεδομένων.
Υπάρχουν projects (open source αρκετά από αυτά) που χρησιμοποιούν το Redis με αυτό τον τρόπο, όπως το RestMQ, το RQ Python job queue και άλλα.

4.4 Leaderboard Listing

Μια ακόμη διασκεδαστική αλλά και πολύ ταιριαστή στο Redis εφαρμογή είναι τα leaderboards. Η ατομικότητα των εντολών και η μεγάλη ταχύτητα με την οποία γίνονται προσθαφαιρέσεις σε τιμές του Redis το καθιστούν πολύ καλή επιλογή για τέτοιες εφαρμογές. Αν για παράδειγμα θέλω να πάρω τις 10 πρώτες τιμές σε ένα sorted set με όνομα myscores χρειάζομαι μια και μονάδικη ατομική εντολή: 'ZRANGE myscores 0 10'.
Χρήση αυτής της δυνατότητας κάνει η Agora Games, δημιουργός μεγάλων videogame τίτλων όπως το Guitar Hero, Tony Hawk series, Portal 2 και άλλα. Το project της πάνω στο Redis είναι open source και βρίσκεται στις αναφορές του report.

4.5 Pub/Sub

Οι εφαρμογές του μοτίβου Pub/Sub (Publish/Subscribe) είναι αμέτρητες. Είναι μια τεχνική ανταλλαγής μηνυμάτων στην οποία οι αποστολείς (publishers) δημοσιοποιούν μηνύματα χωρίς συγκεκριμένο παραλήπτη. Παραλήπτες είναι όσοι επιθυμούν να λαμβάνουν μηνύματα από το προαναφερθέντα publisher, και για αυτό το λόγο κάνουν subscribe σε αυτόν. Αυτή η τεχνική χρησιμοποιείται ευρέως σε μέσα κοινωνικής δικτύωσης (βλ. instagram, youtube κ.α.) αλλά και σε μη παραδοσιακές pub-sub εφαρμογές όπως chat apps, script triggering apps και πολλά άλλα.

5 Ενσωματωμένες Τεχνικές

5.1 Replication

Το Redis χρησιμοποιεί την τεχνική του Replication που στη βάση του είναι μια απλή τεχνική leader-follower (master-slave) replication που επιτρέπει σε κάποιον slave να είναι ακριβές αντίγραφο του master, και λειτουργεί με τρεις βασικούς μηχανισμούς:

- Όταν υπάρχει καλή σύνδεση μεταξύ master και slave, ο master κρατάει τον slave ενήμερο στέλνοντας του συνεχώς ροές απο τις εντολές που εκτελεί ο ίδιος ώστε να είναι ακριβές αντίγραφο του.
- Αν για οποιονδήποτε λόγο χαλάσει η σύνδεση, επιδιώκεται μερικός επανασυγχρονισμός, κατά τον οποίο ο slave προσπαθεί να ανακτήσει τις εντολές που δεν έλαβε κατά τη διάρκεια του downtime.

– Αν δεν επιτευχθεί αυτό το εγχείρημα προχωράει σε ολικό επανασυγχρονισμό, κατά τον οποίο ο master δημιουργεί μέσω ξεχωριστής διεργασίας ένα snapshot όλων των δεδομένων του, το οποίο υιοθετεί ο slave και από εκείνο το σημείο και μετά συνεχίζουν με ροές εντολών.

Η τεχνική του replication υλοποιείται με non-blocking ασύγχρονο τρόπο στην πλευρά του master ο οποίος δεν περιμένει απάντηση από τους slaves ότι έλαβαν τις εντολές του. Υποστηρίζεται όμως και προεραιτική σύγχρονη λειτουργία με τη χρήση της εντολής WAIT.
Ακόμη, το Redis υποστηρίζει πολλαπλούς slaves, αλλά και slaves άλλων slaves.
Γενικά, σαν τεχνική μπορεί να χρησιμοποιηθεί είτε για τον απλό λόγο της ασφάλειας των δεδομένων και του υψηλού availability, όπως και για να κάνουμε off-load κάποιο φόρτο εργασίας από τον master και να αναθέσουμε απλές O(N) εργασίες στους διάφορους slaves.

5.2 Persistence

Το Redis προσφέρει δύο μεγάλες κατηγορίες Persistence, το AOF και το RDB Persistence, καθένα από τα οποία είναι κατάλληλο για διαφορετικές περιστάσεις και προτεραιότητες των χρηστών. Υπάρχει, βέβαια και η δυνατότητα επιλογής No Persistence, ή η χρήση ενός υβρίδιου των δύο τεχνικών. Ας δούμε όμως τις τεχνικές και τα tradeoffs τους:

RDB Persistence
Η τεχνική RDB ουσιαστικά παράγει ένα snapshot των δεδομένων ανά ορισμένα χρονικά διαστήματα και αποθηκεύει αυτό το compact αρχείο στον δίσκο.

Πλεονεκτήματα:

– Το RDB είναι μια συμπαγής αναπαράσταση των αρχείων μας σε ένα χρονικό σημείο, και αυτό το κάνει χρήσιμο για backups.Μπορούμε να αποθηκεύουμε ανά 1 ώρα ένα RDB αρχείο και να έχουμε τις προηγούμενες versions του Redis ανα μία ώρα όσο στο παρελθόν επιθυμούμε.
– Το RDB συμφέρει για ανάκτηση μετά από καταστροφή. Είναι ενα μοναδικό αρχείο το οποίο μπορεί εύκολα να μεταφερθεί σε μακρινά data centers ή clusters υπολογιστών, ακόμα και να κρυπτογραφηθεί για ασφάλεια.
– Η διαδικασία του RDB βελτιστοποιεί την απόδοση του, καθώς το μόνο πράγμα που κάνει η κύρια διεργασία είναι να κάνει ένα fork και να αφήσει την υπόλοιπη δουλειά στη διεργασία-παιδί, συνεχίζοντας κανονικά να εκτελεί εργασίες I/O και μη.
– Το RDB προσφέρει συγκριτικά πιο γρήγορες επανεκκινήσεις από το AOF.

Μειονεκτήματα:

– Υπάρχει μεγάλη πιθανότητα απώλειας δεδομένων που έχουν προστεθεί στη βάση μετά το τελευταίο snapshot και πριν το επόμενο, σε περίπτωση σφάλματος. Φυσικά, τα τελευταία δεδομένα που χάνονται είναι χρονικά πίσω το πολύ όσο και το μέγεθος των διαστημάτων χρόνου που επιλέγουμε για το Persistence.
– Σε χαμηλού βελησνεκούς επεξεργαστές, η διαδικασία του forking για μεγάλα datasets μπορεί να οδηγήσει σε ένα μικρό χρονικό downtime.

AOF Persistence
Η τεχνική AOF είναι μια τεχνική logging που καταγράφει με χρονική σειρά οποιοδήποτε write command του server με σκοπό την βαθμιαία επανάκτηση του dataset σε περίπτωση σφάλματος.

Πλεονεκτήματα:

- Προσφέρονται διαφορετικές πολιτικές fsync, και το default του ενός δευτερολέπτου εγγυάται μέχρι ενός δευτερολέπτου πριν απώλεια σε περίπτωση σφάλματος.
- Το AOF log είναι append-only, δηλαδή επιτρέπει μόνο πρόσθεση εντολών και όχι σβήσιμο ή αλλαγή ήδη γραμμένων εντολών. Έτσι αποφεύγονται περιπτώσεις διαστρέβλωσης, και ακόμα και στην περίπτωση μισών εντολών, μπορούν να ανακτηθούν εύκολα με το εργαλείο redis-check-aof.
- Όταν το AOF log γίνει πολύ μεγάλο γίνεται αυτόματο rewrite και συνέχεια από νέο αρχείο εγγραφής των εντολών, χωρίς κάποιο blocking.
- Το αρχείο του AOF είναι εύκολα αναγνώσιμο και επεξεργάσιμο από τον χρήστη.

Μειονεκτήματα:

- Τα αρχεία AOF είναι μεγαλύτερα από τα RDB.
- Τείνει να είναι πιο αργό από το RDB ανάλογα και με την fsync πολιτική, καθώς κάνει παραδείγματος χάρην το logging κάθε δευτερόλεπτο σε σύγκριση με τα 5 λεπτά του RDB.
- Έχουν παρατηρηθεί σπάνια bugs τα οποία δεν εμφανίζονται στο RDB Persistence, ωστόσο είναι μόνο σε testing environments των μηχανικών του Redis, και δεν έχουν αναφερθεί από πραγματικούς πελάτες.

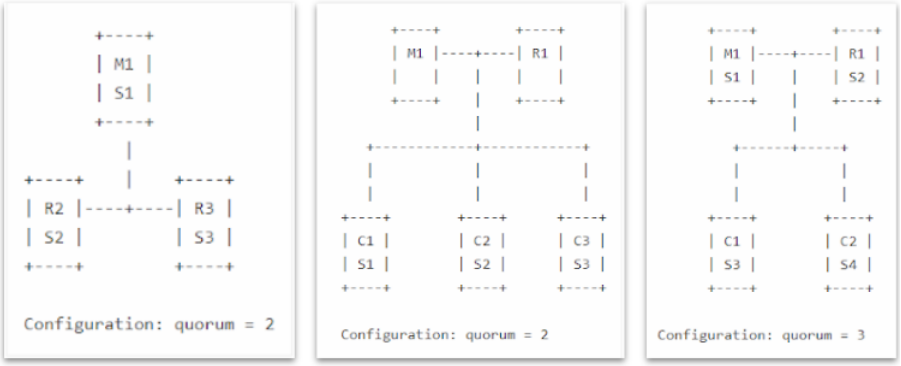
Η γενική κρίση λέει ότι είναι μια καλή πρακτική να εφαρμόζονται και οι δύο μέθοδοι ταυτόχρονα για να είναι όσο πιο ασφαλής και consistent γίνεται η βάση δεδομένων μας.

5.3 Availability - Redis Sentinel

Το Redis προσφέρει high availability χρησιμοποιώντας το Redis Sentinel. Πρακτικά αυτό σημαίνει ότι ένα Redis deployment μπορεί να ανακτά μόνο του την κατάσταση του σε περίπτωση κάποιας καταστροφής χωρίς να απαιτείται ανθρώπινη παρεμβολή. Συγκεκριμένα, το Redis Sentinel προσφέρει δυνατότητες:

- Monitoring
- Notification
- Automatic Failover
- Configuration Provider

Το Sentinel είναι ένα κατανεμημένο σύστημα στη φύση του. Είναι σχεδιασμένο να τρέχει με πολλαπλά instances που συνεργάζονται μεταξύ τους, και εφόσον παρατηρήσουν σφάλμα πρέπει να συμφωνήσουν στο σφάλμα για να ληφθεί κάποια δράση, ενώ λειτουργούν το καθένα αυτόνομα.



Στην εικόνα μπορούμε να δούμε σε διαφορετικά master-slave-client deployments πώς μπορούμε να τοποθετήσουμε τα sentinels και πόσο πρέπει να είναι το quorum (ελάχιστος αριθμός sentinels που συμφωνούν ώστε να ληφθεί κάποια δράση) στην εκάστοτε περίπτωση.

5.4 Partitioning-Sharding

Σε κάποιες περίπτωσεις όπως όταν χειριζόμαστε μεγάλα δεδομένα, χρειάζεται να κατευθυνθούμε προς clusters υπολογιστών και να μοιράσουμε τα δεδομένα μας σε αυτούς με τεχνικές partitioning.
Τα δύο πιο γνωστά partitioning schemes είναι το απλό **range partitioning**, κατά το οποίο επιλέγουμε ένα range N και μοιράζονται με τη σειρά ανά N τα δεδομένα στα instances, και το **hash partitioning** κατά το οποίο κάθε κλείδι μετατρέπεται σε hash και το υπόλοιπο της διαίρεσης με το πλήθος των διαθέσιμων instances καθορίζει σε πιο instance θα αντεθεί το κλειδί.
Οι υλοποιήσεις του partitioning είναι τριών τύπων:

- **Client side partitioning**: Ο client υποδεικνύει από ποιό instance θα γίνει το διάβασμα ή σε ποιό θα γίνει το γράψιμο.
- **Proxy assisted partitioning**: Ο client κάνει το ερώτημα σε κάποιο proxy το οποίο προωθεί με τη σειρά του την ενέργεια του client στο σωστό instance. Το Redis και το Memcached χρησιμοποιούν το Twemproxy για αυτή τη διαδικασία.
- **Query routing**: Ο client κάνει το ερώτημα προς ένα τυχαίο instance, το οποίο με τη σειρά του το προωθεί στο σωστό instance. Το Redis Cluster κάνει την παραπάνω διεργασία με ένα είδος υβριδικού Query routing, ανακατευθύνοντας τον client στο σωστό instance.

6 Σχεσιακές Βάσεις Δεδομένων vs. Redis

Έχοντας χρησιμοποιήσει και τους δύο τύπους βάσεων δεδομένων μπορώ να πραγματοποιήσω μια σύντομη σύγκριση μεταξύ των δύο.
Ένα μεγάλο πλεονέκτημα του Redis είναι η αποθήκευση σύνθετων δομών δεδομένων

και η ασάφεια που το καταλαμβάνει. Στις παραδοσιακές σχεσιακές βάσεις δεν μπορώ να αποθηκεύσω για παράδειγμα μια γραμμική λίστα δεδομένων, και θα πρέπει να βάλω τα δεδομένα με τη σειρά σε ένα πίνακα τον οποίο θα συσχετίσω με ένα άλλο πίνακα που μια τιμή του θα χαρακτηρίζει την λίστα. Η διαδικασία αυτή εισάγει άχρηστη πολυπλοκότητα στο πρόβλημα που καλείται να λύσει κάποιος και για αυτό το λόγο θα προτιμούσα το Redis για την αναπαράσταση σύνθετων δομών και όχι απλών strings,chars ή integers.
Ένα άλλο πλεονέκτημα είναι η μεγάλη διαφορά ταχύτητας. Σαν in-memory βάση, το Redis έχει μεγάλο πλεονέκτημα σε ταχύτητα απέναντι στις disk-storing σχεσιακές βάσεις δεδομένων.
Για τον ίδιο λόγο (in-memory) ,όμως, θα επέλεγα μια βάση που αποθηκεύει στο δίσκο για πολύ μεγάλα datasets. Το Redis, έχοντας σαν τόπο αποθήκευσης την RAM έχει μικρό όριο στο μέγεθος των δεδομένων που μπορεί να εξυπηρετήσει.
Τέλος, ένας ακόμη λόγος που θα προτιμούσα μια σχεσιακή βάση σε σύγκριση με το Redis, είναι λόγω της δυσκολίας (αν και παρέχεται η βάση) της υλοποίησης του clustering σε Redis instances.

7 Πηγές

Στις αναφορές του report παρέχονται σύνδεσμοι προς υλικό από το οποίο άντλησα πληροφορίες και γνώσεις. Ακόμη, κατά τη διάρκεια υλοποίησης της εφαρμογής μου συνηδειτοποίησα το βάθος του documentation που παρέχει το redis, και το πόσο μπορεί να ωθήσει κάποιον αρχάριο στην πλατφόρμα στο να χτίσει με αυτή χρήσιμες και ενδιαφέρουσες εφαρμογές.

Αναφορές

1. Redis.io official site, https://redis.io/
2. Joe Engel : Top 5 Redis use cases, https://www.objectrocket.com/blog/how-to/top-5-redis-use-cases/
3. Techstack.io : Who is using Redis? , https://techstacks.io/tech/redis
4. Agora Games: leaderboard project, https://github.com/agoragames/leaderboard
5. Twitter: Twemproxy, https://github.com/twitter/twemproxy
6. Redis check AOF tool, https://www.mankier.com/1/redis-check-aof