Κατανεμημένα Συστήματα
Υλοποίηση ToyChord

Ιάσονας Νικολάου
AM: 03116129

Μιλτιάδης Στούρας
AM: 03116022

Χαρίτων Χαριτωνίδης
AM: 03116694

Ακ. Έτος 2020-2021 - Ροή Υ

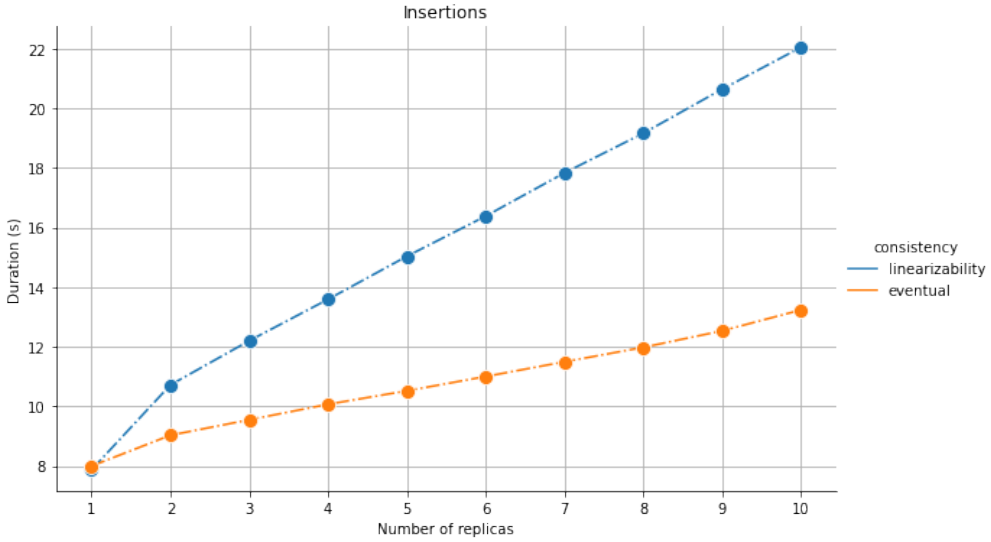
Εισαγωγή

Στην παρούσα εργασία υλοποιήσαμε μια απλουστευμένη έκδοση του πρωτοκόλλου Chord. Συγκεκριμένα, δεν έχουμε μεριμνήσει για αποτυχίες κόμβων και δεν έχουμε υλοποιήσει finger tables.
Υλοποιήσαμε το πρωτόκολλο σε Python και χρησιμοποιήσαμε REST API interfaces για την επικοινωνία των κόμβων (μεταξύ τους και με τους clients).
Για το local development υλοποιήσαμε ένα framework που μας επιτρέπει να κάνουμε deploy κόμβους και όλες τις υπόλοιπες λειτουργίες (insert, delete, query, κλπ), καθώς και τοπικό έλεγχο της τοπολογίας του δικτύου και των replicas.
Επίσης υλοποιήσαμε ένα web application για ευκολότερη αλληλεπίδραση και εποπτεία τού δικτύου των κόμβων.

Πειράματα

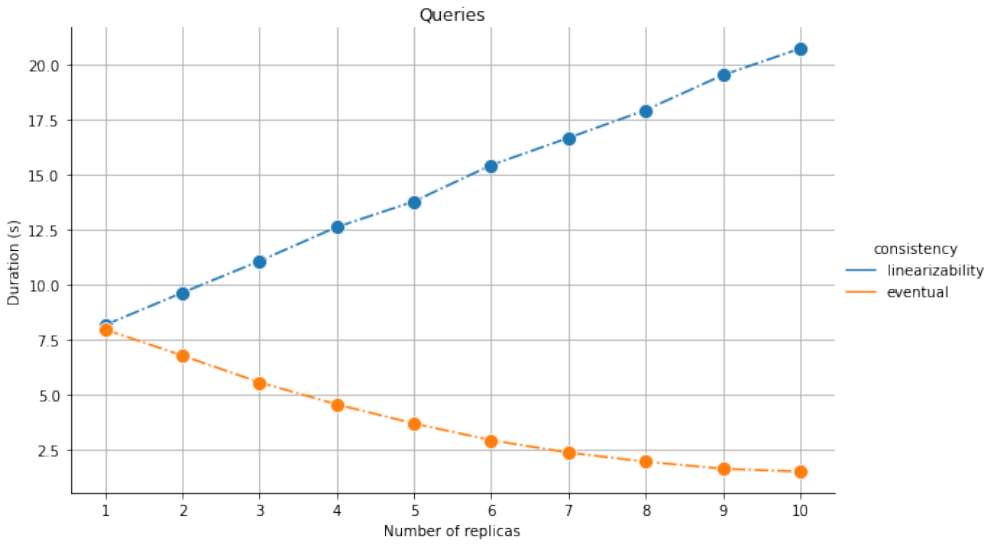
Αυτοματοποιήσαμε την διεξαγωγή πειραμάτων και τρέξαμε τα πειράματα της εργασίας για όλες τις τιμές του k από 1 μέχρι και 10 και για τους 2 τύπους συνέπειας. Ακόμα, επειδή κάθε operation (insert/request/delete) στέλνεται κάθε φορά σε έναν τυχαίο κόμβο του ToyChord, πραγματοποιήσαμε κάθε πείραμα 5 φορές για κάθε setup, ώστε να μειώσουμε την επίδραση της τυχαιότητας στα αποτελέσματα.

Πείραμα 1ο: Batch Insertions



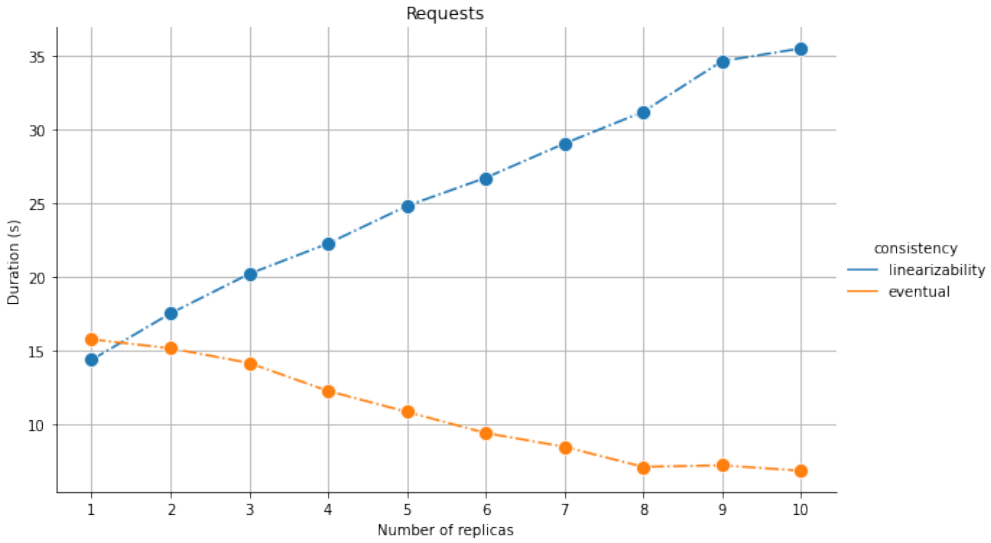
Παρατηρούμε πως καθώς αυξάνεται ο αριθμός των replicas, αυξάνεται και ο χρόνος που χρειάζεται για να εκτελεστεί ο ίδιος αριθμός από insertions, δηλαδή μειώνεται το insertion throughput. Αυτό συμβαίνει καθώς κάθε κόμβος διατηρεί περισσότερα αρχεία και ανά πάσα στιγμή είναι πιο πιθανό να γίνεται update κάποιου αρχείου και η δομή των files να είναι κλειδωμένη. Ο ρυθμός αύξησης του χρόνου ως προς τον αριθμό των replicas είναι μεγαλύτερος στην περίπτωση του linearizability, καθώς εχουμε περισσότερα blocking operations.

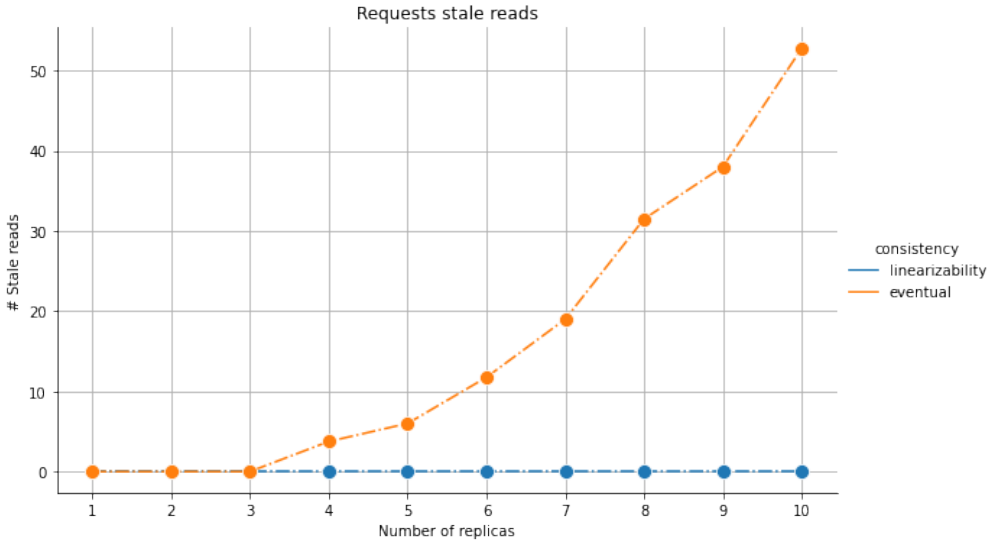
Πείραμα 2ο: Batch queries



Όπως αναμέναμε, το eventual consistency έχει μεγαλύτερο query throughput από το linearizability, και μάλιστα καθώς αυξάνεται ο αριθμός των replicas στο δίκτυο αυξάνεται και το throughput γιατί κατά μέσο όρο χρειαζόμαστε λιγότερα βήματα για να βρούμε κάποιον κόμβο που κατέχει αντίγραφο του αρχείου που αναζητούμε. Η ακραία περίπτωση είναι το $k = 10$ όπου όλοι οι κόμβοι έχουν όλα τα κλειδιά και πράγματι παρατηρούμε το μεγαλύτερο throughput.

Πείραμα 3ο: Mixed operations





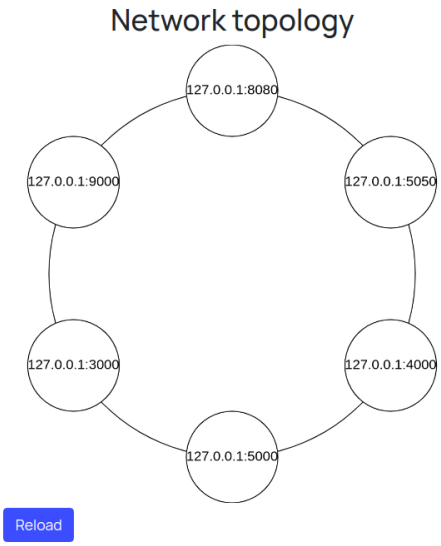
Όπως είχαμε δει και από τα δύο πρώτα tests, το eventual consistency έχει σημαντικά μεγαλύτερο throughput, ωστόσο μπορεί να συναντήσουμε stale reads. Συγκεκριμένα, κατά την διεξαγωγή του πειράματος, κρατούσαμε τοπικά τις ενημερωμένες τιμές που θα πρέπει να έχουν τα αρχεία, σύμφωνα με τα insertions που είχαμε κάνει, και όταν πραγματοποιούσαμε ένα query συγκρίναμε την απάντηση του δικτύου με τις ενημερωμένες τιμές που διατηρούσαμε. Όπως περιμέναμε, στο linearizability δεν παρατηρήσαμε κανένα stale read για καμία τιμή του k. Αντίθετα, στην περίπτωση του eventual consistency παρατηρήσαμε αρκετά stale reads, τα οποία αυξάνονταν όσο αυξανόταν και το k. Αυτό συμβαίνει διότι όταν αυξάνεται ο αριθμός των replicas, χρειάζεται περισσότερος χρόνος για να ενημερωθούν όλοι ενώ ταυτόχρονα εμείς βρίσκουμε σε ακόμα μικρότερο χρόνο κάποιον κόμβο που έχει κάποια τιμή για το κλειδί που ζητάμε.
Οι δύο τεχνικές συνέπειας που υλοποιήσαμε μας δίνουν την δυνατότητα να κάνουμε trade-offs μεταξύ του throughput και του ποσοστού των stale reads.

Testing

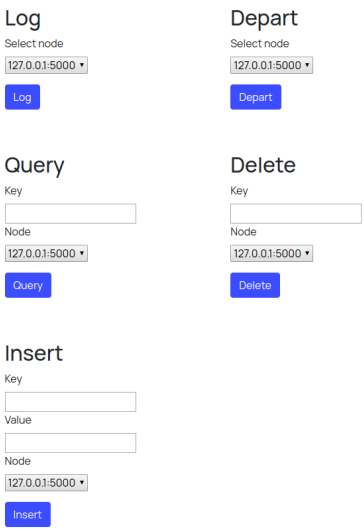
Δημιουργήσαμε unit tests για κάποιες βασικές λειτουργίες που χρειαζόμαστε στην υλοποίηση των κόμβων.
Επίσης κάναμε end-to-end automated tests κατά τα οποία ”σηκώνονται” servers τοπικά, γίνονται εισαγωγές, διαγραφές κλειδιών, αποχωρήσεις κόμβων και σε κάθε βήμα ελέγχονται αυτόματα τα κλειδιά και τα αντίγραφα στους κόμβους, καθώς και η τοπολογία του δικτύου.

Web GUI application

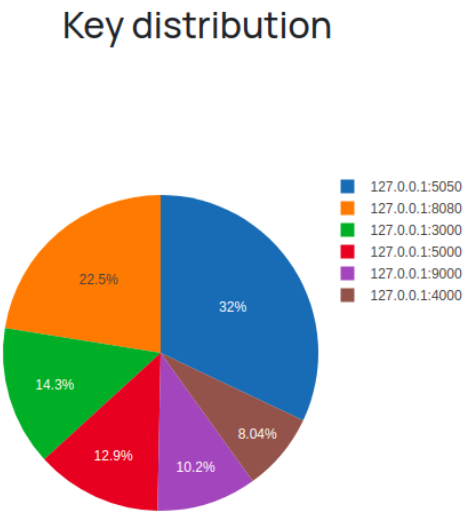
Για το web application χρησιμοποιήσαμε κυρίως react.js και bootstrap. Το web application υποστηρίζει όλες τις λειτουργίες τού ToyChord και την επιπλέον λειτουργία Log η οποία δέχεται ως παράμετρο έναν κόμβο(ip, port) και επιστρέφει όλα τα αρχεία και τα replicas που αυτός περιέχει. Ακόμη, απεικονίζουμε την τοπολογία τού δικτύου καθώς και το ποσοστό των κλειδιών τού δακτυλίου που ελέγχονται από κάθε κόμβο.



(α′) Network Topology



(β′) Supported Actions



(α′) Keys per node

(β′) Log node 127.0.0.1:5000