ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Κατανεμημένα Συστήματα

Ονόματα Ομάδας:

Ασημάκη Γεωργία Γρηγορία (Α.Μ. : 03116197)
Ζάρρας Ιωάννης (Α.Μ.: 03116082)
Παπαγεωργίου Νικολέττα Ευσταθία (Α.Μ.: 03116648)

Πειράματα:

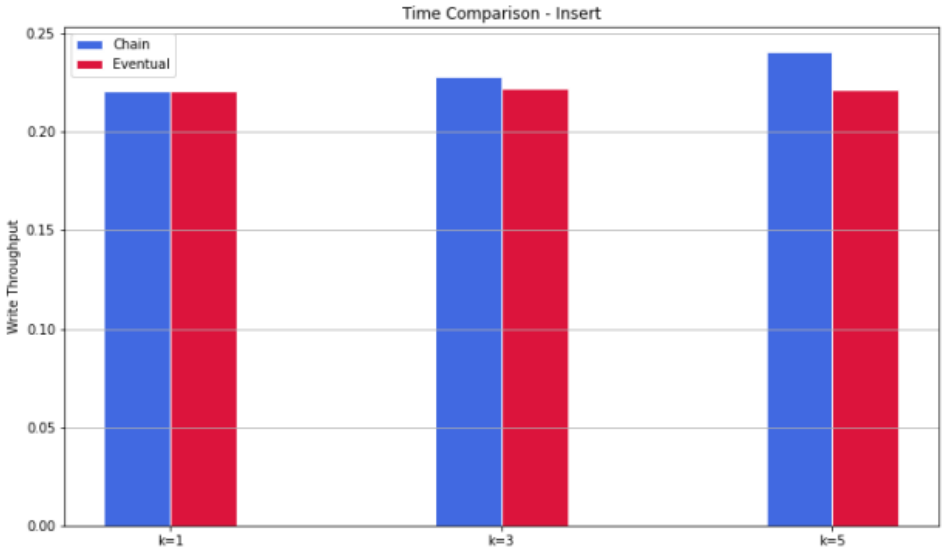
Αναπτύξαμε το ToyChord στη γλώσσα Python αξιοποιώντας το Flask framework. Κάνοντας χρήση των πόρων που μας δόθηκαν στον Okeanos . Συγκεκριμένα:

Linearizability: Θα πρέπει να υπάρχουν ισχυρές εγγυήσεις ότι όλα τα replicas έχουν πάντα την ίδια τιμή για κάθε κλειδί και ότι κάθε query θα επιστρέφει πάντα την πιο πρόσφατη τιμή που έχει γραφτεί. Επιλέξαμε να υλοποιήσουμε chain replication.

Chain replication: Σε αυτήν την περίπτωση ένα write ξεκινά πάντα από τον πρωτεύοντα κόμβο που είναι υπεύθυνος για ένα κλειδί και προχωρά με τη σειρά στους k-1 υπόλοιπους που έχουν αντίγραφα. Ο τελευταίος κόμβος στη σειρά επιστρέφει το αποτέλεσμα του write. Ένα read αντιθέτως πρέπει να διαβάζει την τιμή από τον τελευταίο κόμβο στη σειρά.

Eventual consistency: οι αλλαγές διαδίδονται lazily στα αντίγραφα. Αυτό σημαίνει ότι ένα write θα πηγαίνει στον πρωτεύοντα κόμβο που είναι υπεύθυνος για το συγκεκριμένο κλειδί και ο κόμβος αυτός θα επιστρέφει το αποτέλεσμα του write. Στη συνέχεια θα φροντίζει να στείλει τη νέα τιμή στους k-1 επόμενους κόμβους. Ένα read θα διαβάζει από οποιονδήποτε κόμβο έχει αντίγραφο του κλειδιού που ζητά (με κίνδυνο να διαβάσει stale τιμή).

- **Εισάγαμε σε ένα DHT με 10 κόμβους όλα τα κλειδιά που βρίσκονται στο αρχείο insert.txt με k=1 (χωρίς replication), k=3 και k=5 και με linearizability και eventual consistency.**

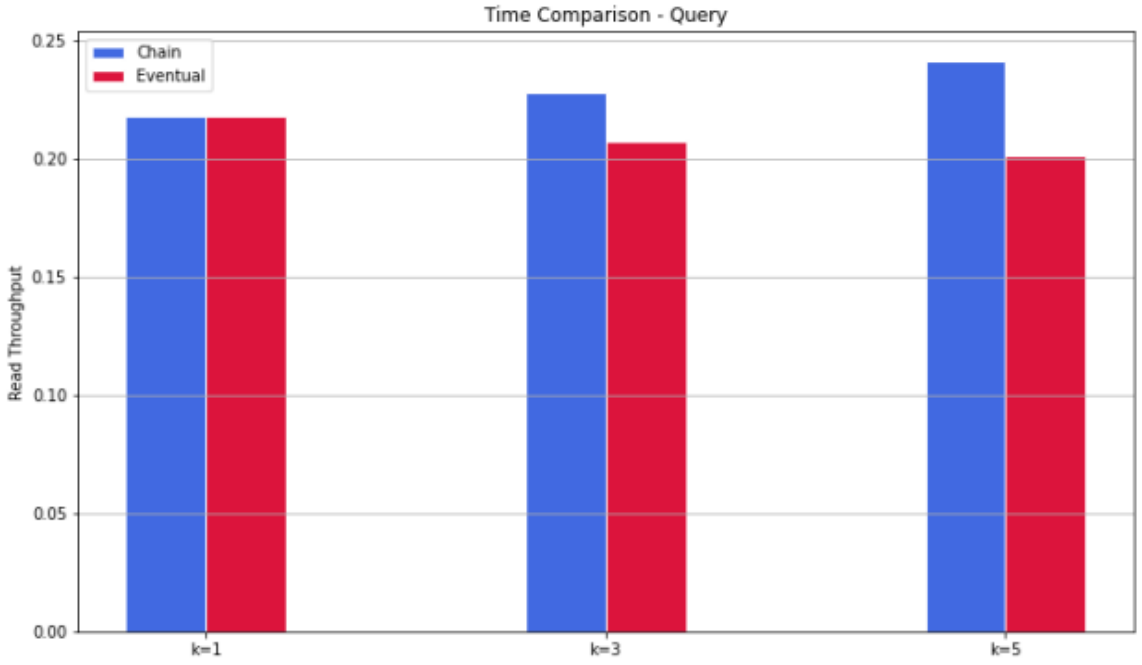


Συμπεράσματα:

Όπως είναι λογικό, στην περίπτωση που δεν έχουμε replication (δηλαδή k = 1) παρατηρούμε ότι δεν υπάρχει διαφορά ανάμεσα στις δύο συνέπειες(linearizability και eventual consistency.
Αντίθετα, όταν έχουμε replication (δηλαδή k > 1) παρατηρούμε ότι το eventual consistency είναι γρηγορότερο σε σχέση με το linearizability. Αυτό συμβαίνει γιατί στην περίπτωση του chain replication (του linearizability) το write επιστρέφεται από το τελευταίο replicas της σειράς. Ακόμα, παρατηρούμε ότι αυξάνοντας το k (αριθμό replicas) αυξάνεται και το throughput του chain replication(linearizability) αφού αυξάνεται η αλυσίδα και τα αντίγραφα. Τέλος, παρατηρούμε σχεδόν σταθερή τιμή στο throughput του eventual consistency και όποια μικρή διαφορά οφείλεται στην τυχαία επιλογή του κόμβου από τον οποίο ξεκινάει το κάθε πείραμα.

- ***Για τα 6 διαφορετικά setups του προηγούμενου ερωτήματος, διαβάσαμε όλα τα keys που βρίσκονται στο αρχείο query.txt και καταγράψαμε το read throughput. Τα queries ξεκινούν κάθε φορά από τυχαίο κόμβο. Ο κόμβος ελέγχει αν έχει το κλειδί ή αντίγραφό του, αλλιώς προωθεί το query στον επόμενο.***

Παρακάτω παρουσιάζονται τα αποτελέσματα για τις 6 παραπάνω περιπτώσεις.



Συμπεράσματα:
Ομοίως με τα παραπάνω πειράματα παρατηρούμε ότι στην περίπτωση που δεν έχουμε replicas (k = 1) δεν υπάρχει διαφορά ανάμεσα στις δύο συνέπειες. Ωστόσο, αυξάνοντας το k (αριθμό replicas) παρατηρούμε ότι αυξάνει το throughput του chain

replication (linearizability) ενώ το eventually consistency παραμένει πάλι γρηγορότερο αφού επιστρέφει το πρώτο αντίγραφο που θα εντοπίσει και όχι το τελευταίο στη σειρά (όπως κάνει το chain replication) .Τέλος, έχουμε πάλι τυχαία επιλογή του κόμβου που κάνουμε insert αλλά έχουμε και περισσότερα αντίγραφα άρα αυξάνεται η πιθανότητα να εντοπίσουμε κάποιο replicas και να το επιστρέψουμε και έτσι όσο αυξάνεται το k μειώνεται το throughput του eventual consistency .

- **Για DHT με 10 κόμβους και k=3, εκτελέσαμε τα requests του αρχείου requests.txt. Στο αρχείο αυτό η πρώτη τιμή κάθε γραμμής δείχνει αν πρόκειται για insert ή query και οι επόμενες τα ορίσματά τους.**

Συμπεράσματα:

Όπως είναι λογικό, οι απαντήσεις των queries διαφέρουν λίγο σε περίπτωση linearization και eventual consistency. Πιο fresh τιμές δίνει το chain replication (linearizability) αφού στην περίπτωση του eventual consistency (που επιστρέφει την πρώτη τιμή που θα συναντησει) υπάρχει πιθανότητα το αντίγραφο που θα επιστρέψει να μην έχει ανανεωθεί ακόμα δηλαδή να επιστρέψει μια stale τιμή. Κάνοντας τα πειράματα παρατηρούμε ότι με τους 10 κόμβους δεν παρατηρήθηκε κάποια αλλαγή σε αυτά που επιστρέφουν τα queries. Αντίθετα, στους 3 κόμβους είναι εμφανή τα παραπάνω που αναφέρθηκαν και έχουμε fresh τιμές στο chain replication(linearizability).

Παρακάτω βλέπουμε τη διαφορά που προκαλεί η αντιμετώπιση των queries από τα δύο διαφορετικά είδη consistency, όπως αυτή εμφανίστηκε στο πείραμα με τους 3 κόμβους. Παρουσιάζουμε το αποτέλεσμα στον ίδιο κόμβο, για τα δύο διαφορετικά consistencies:

[('123b432ae2b48b42b9efbd1b43245fbd37e43e6d', 'Respect', '589', '1'), ('09f22dd8eaeb6b38569b06a8045e3e2d6a938e44', "What's_Going_On", '592', '1'), ('0b6b2aae684c8cdfb6e41977bba1e2558a8a49ff', 'Hey_Jude', '598', '1'), ('8673facf0410d681b6e61a9904daae141c216bef', 'Like_a_Rolling_Stone', '600', '2')]

[('8673facf0410d681b6e61a9904daae141c216bef', 'Like_a_Rolling_Stone', '600', '2'), ('123b432ae2b48b42b9efbd1b43245fbd37e43e6d', 'Respect', '589', '1'), ('09f22dd8eaeb6b38569b06a8045e3e2d6a938e44', "What's_Going_On", '592', '1'), ('0b6b2aae684c8cdfb6e41977bba1e2558a8a49ff', 'Hey_Jude', '598', '1')]