ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

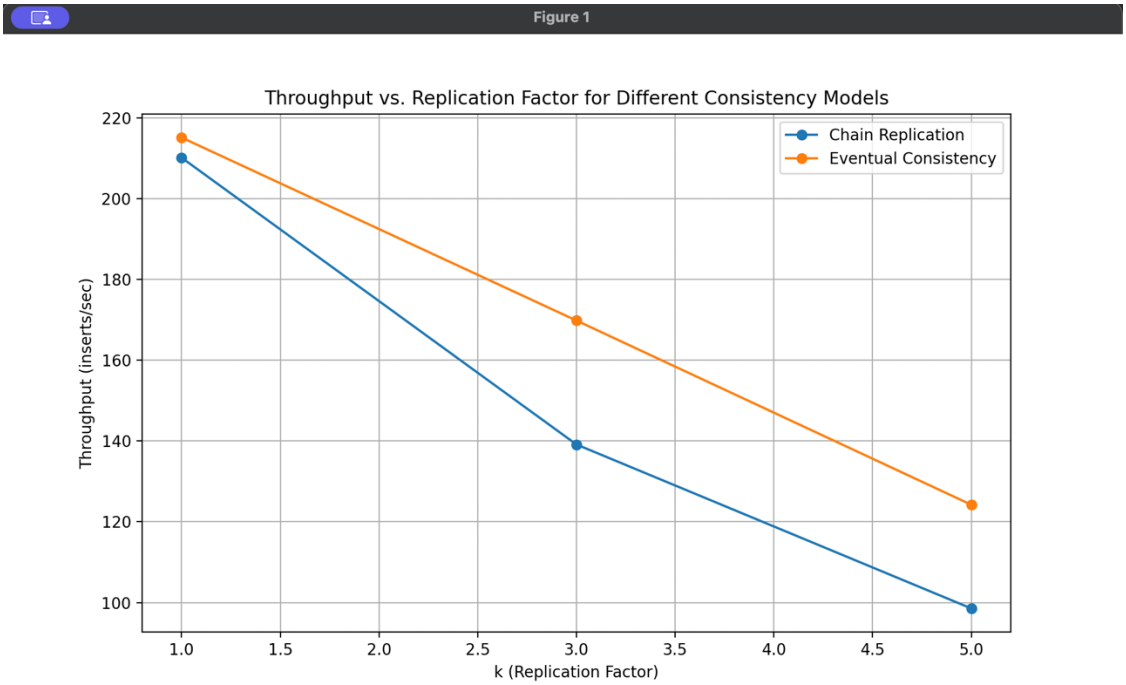
Εξαμηνιαία Εργασία στο μάθημα «Κατανεμημένα Συστήματα»

ΣΠΟΥΔΑΣΤΗΣ: ΓΙΑΝΝΑΚΟΠΟΥΛΟΣ ΣΤΕΦΑΝΟΣ ΑΜ.:03120829
ΣΠΟΥΔΑΣΤΡΙΑ: Δήμητρα Τσέγκου, Α.Μ.: 03120633

Διδάσκοντες: Νεκτάριος Κοζύρης, Καθηγητής ΕΜΠ
Κατερίνα Δόκα, Μεταδιδακτορική Ερευνήτρια

Αθήνα, Μάρτιος 2025

<u>ΠΕΙΡΑΜΑ 1ο:</u> Εισάγετε σε ένα DHT με 10 κόμβους όλα τα κλειδιά που βρίσκονται στα αρχεία insert_n.txt (όπου n το id του κόμβου που έχει τα τραγούδια που περιέχονται στο αρχείο) με k=1 (χωρίς replication), k=3 και k=5 και με linearizability και eventual consistency (δλδ 6 πειράματα) και καταγράψτε το write throughput του συστήματος (πόσο χρόνο πήρε η εισαγωγή των κλειδιών προς τον αριθμό τους). Τα inserts των 10 αρχείων θα ξεκινούν ταυτόχρονα από τους 10 κόμβους του συστήματος.



Τι συμβαίνει με το throughput όταν αυξάνεται το k στις δύο περιπτώσεις consistency;

Όταν αυξάνεται ο αριθμός των αντιγράφων k, τόσο στο **chain replication** όσο και στο **eventual consistency**, παρατηρείται μείωση του throughput (δηλαδή των εγγραφών ανά δευτερόλεπτο). Ωστόσο, ο τρόπος και ο ρυθμός μείωσης διαφέρει λόγω του τρόπου με τον οποίο λειτουργεί κάθε μοντέλο:

1.Chain Replication

→ Στο chain replication, οι κόμβοι οργανώνονται σε μία αλυσίδα (chain). Για να ολοκληρωθεί μια εγγραφή, το αίτημα περνάει σειριακά από όλους τους κόμβους, ξεκινώντας από τον πρώτο και καταλήγοντας στον τελευταίο.

→ Όταν αυξάνεται ο αριθμός των κόμβων k, μεγαλώνει η «αλυσίδα» και κάθε εγγραφή πρέπει να «ταξιδέψει» μέσα από περισσότερους κόμβους. Αυτό επιμηκύνει τον χρόνο ολοκλήρωσης κάθε εγγραφής, αφού περιμένουμε και τις επιβεβαιώσεις κατά μήκος ολόκληρης της αλυσίδας.

→ Ως αποτέλεσμα, το συνολικό σύστημα επιβαρύνεται περισσότερο και ο ρυθμός (throughput) εγγραφών μειώνεται αισθητά καθώς αυξάνεται το k.

2. Eventual Consistency

→ Στο eventual consistency, η ενημέρωση των αντιγράφων γίνεται με πιο χαλαρούς περιορισμούς συγχρονισμού. Συνήθως, ο κύριος κόμβος (ή οποιοσδήποτε κόμβος λάβει την εγγραφή) μπορεί να απαντήσει άμεσα στον πελάτη ότι η εγγραφή ολοκληρώθηκε και, στη συνέχεια, διαχέει την ενημέρωση στους υπόλοιπους κόμβους ασύγχρονα.

→ Παρότι και σε αυτό το μοντέλο, καθώς αυξάνεται το k, απαιτείται να σταλούν περισσότερες ενημερώσεις (ένα μήνυμα προς κάθε νέο αντίγραφο), αυτές μπορούν να γίνουν σε μεγάλο βαθμό ταυτόχρονα ή

ασύγχρονα, χωρίς να απαιτούνται πλήρεις σειριακές επιβεβαιώσεις από όλους τους κόμβους πριν το σύστημα θεωρήσει την εγγραφή έγκυρη.

→ Έτσι, το overhead αναπαραγωγής, αν και αυξάνεται, δεν «φρενάρει» τόσο έντονα το σύστημα όσο στο chain replication. Γι' αυτόν τον λόγο, το throughput μειώνεται με πιο ήπιο ρυθμό.

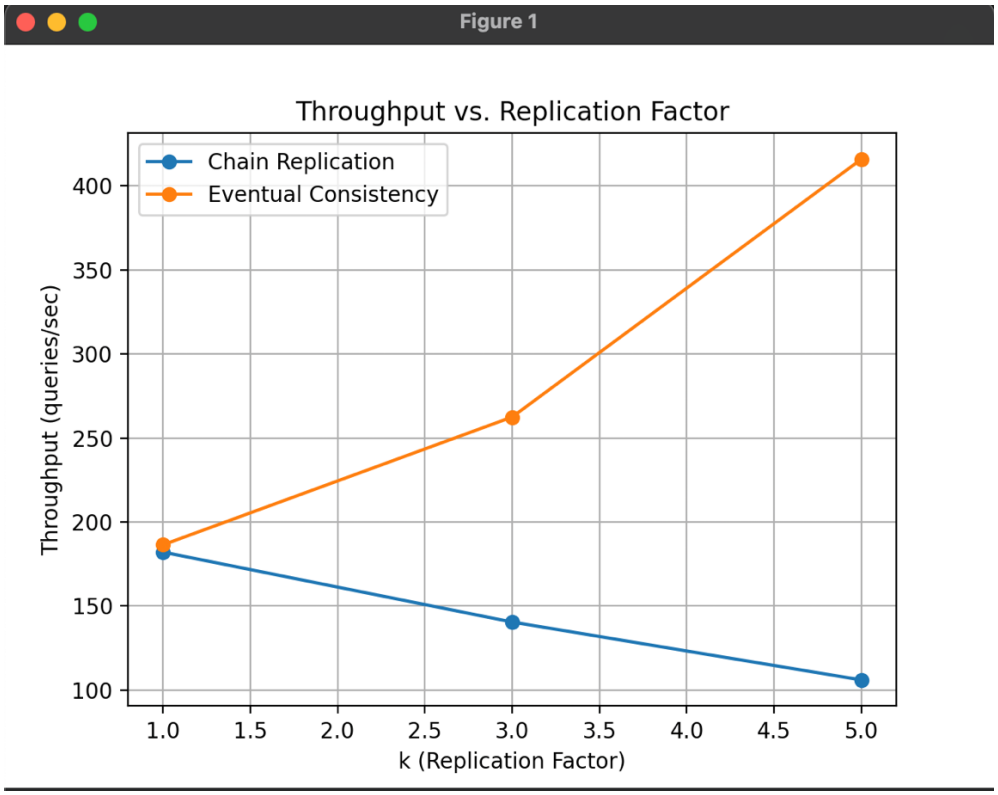
3.Γιατί μειώνεται το throughput και στις δύο περιπτώσεις;

→ Κάθε επιπλέον αντίγραφο (αύξηση του k) συνεπάγεται επιπλέον κόστη επικοινωνίας, αποστολής δεδομένων και ενημερώσεων/επιβεβαιώσεων.

→ Στο **chain replication**, αυτά τα κόστη προστίθενται ουσιαστικά σειριακά (η εγγραφή περνάει υποχρεωτικά από όλους τους κόμβους), οπότε η μείωση του throughput είναι πιο εμφανής.

→ Στο **eventual consistency**, το κόστος προστίθεται κυρίως λόγω περισσότερων μηνυμάτων, αλλά υπάρχει μεγαλύτερη ελευθερία στο πώς και πότε αυτά τα μηνύματα αποστέλλονται/επιβεβαιώνονται. Αυτό επιτρέπει υψηλότερο βαθμό παράλληλης επεξεργασίας, με αποτέλεσμα ο ρυθμός μείωσης του throughput να είναι πιο αργός.

ΠΕΙΡΑΜΑ 2ο: Για τα 6 διαφορετικά setups του προηγούμενου ερωτήματος, διαβάστε όλα τα keys που βρίσκονται στα αρχεία query_n.txt και καταγράψτε το read throughput. Τα queries ξεκινούν ταυτόχρονα από τον κόμβο που υποδεικνύει το n του κάθε αρχείου. Ο κόμβος ελέγχει αν έχει το κλειδί ή αντίγραφό του, αλλιώς προωθεί το query στον επόμενο.



Στο πείραμα αυτό μετράμε το **read throughput** (δηλαδή τον ρυθμό αναγνωστικών αιτημάτων/δευτερόλεπτο) σε ένα σύστημα που διαθέτει αριθμό αντιγράφων (replicas) k και μπορεί να λειτουργεί σε διαφορετικά μοντέλα συνοχής (consistency).

1. Chain Replication (k=3, k=5)

→ Για k=3, ο χρόνος ήταν περίπου **3.59 sec** για 500 reads (throughput **140.50 reads/sec**).
→ Για k=5, ο χρόνος ήταν περίπου **4.73 sec** για 500 reads (throughput **106.04 reads/sec**).

Στο chain replication οι αναγνώσεις (reads) συνήθως σερβίρονται από έναν συγκεκριμένο κόμβο την tail της αλυσίδας, ώστε να εξασφαλιστεί ότι τα δεδομένα είναι πλήρως ενημερωμένα. Όταν αυξάνεται ο αριθμός των αντιγράφων (k), η αλυσίδα μεγαλώνει και συνεπώς ο συγχρονισμός και η διαχείριση γίνονται πιο περίπλοκα. Επιπλέον, αν όλα τα reads κατευθύνονται σε έναν (ή λίγους) κόμβους, μπορεί να προκύψει bottleneck. Έτσι, όσο αυξάνεται το k, **μειώνεται** το συνολικό throughput των αναγνώσεων.

2. Eventual Consistency (k=3, k=5)

→ Για k=3, ο χρόνος ήταν περίπου **1.94 sec** για 500 reads (throughput **262.48 reads/sec**).
→ Για k=5, ο χρόνος ήταν περίπου **1.22 sec** για 500 reads (throughput **415.90 reads/sec**).

Στο eventual consistency, οι ενημερώσεις διαχέονται στους υπόλοιπους κόμβους με καθυστέρηση (lazily), ενώ οι αναγνώσεις μπορούν να εξυπηρετούνται από **οποιονδήποτε** κόμβο έχει αντίγραφο του δεδομένου. Αυτό έχει ως αποτέλεσμα, όσο αυξάνεται ο αριθμός κόμβων (k), να είναι διαθέσιμα **περισσότερα** σημεία εξυπηρέτησης, με ταυτόχρονη επεξεργασία πολλών αιτημάτων. Έτσι, το σύστημα μπορεί να χειριστεί περισσότερα read requests ανά δευτερόλεπτο (αυξημένο throughput).

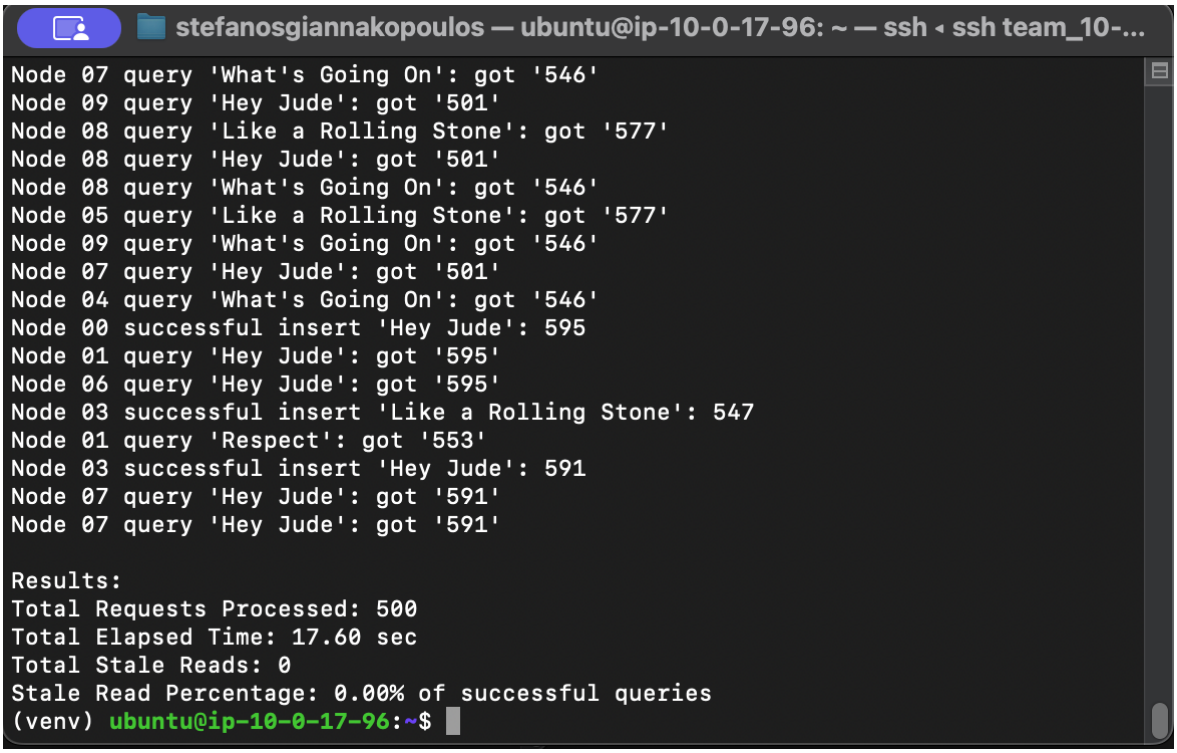
3. Συνολικό Συμπέρασμα

- **Chain Replication:** Προσφέρει ισχυρότερη συνοχή (διαβάζεις πάντα τα πιο πρόσφατα δεδομένα), όμως η αύξηση του k επιβαρύνει σημαντικά το σύστημα, μειώνοντας το read throughput.
- **Eventual Consistency:** Το μοντέλο είναι πιο "χαλαρό" όσον αφορά τη συνοχή (risk of stale reads), αλλά κάθε κόμβος μπορεί να εξυπηρετεί αναγνώσεις. Έτσι, όσο περισσότερα αντίγραφα υπάρχουν, τόσο περισσότερους κόμβους έχουμε για τα reads, γεγονός που αυξάνει το throughput.

ΠΕΙΡΑΜΑ 3ο: Για DHT με 10 κόμβους και k=3, εκτελέστε τα requests των αρχείων requests_n.txt. Στο αρχείο αυτό η πρώτη τιμή κάθε γραμμής δείχνει αν πρόκειται για insert ή query και οι επόμενες τα ορίσματά τους.

1.Για chain replication και k=3:

Στο Chain Replication οι εγγραφές πραγματοποιούνται στον head node και προωθούνται σειριακά σε όλους τους κόμβους μέχρι τον tail node. Οι queries εκτελούνται αποκλειστικά στον tail node, διασφαλίζοντας ότι επιστρέφεται πάντα η πιο πρόσφατη τιμή.

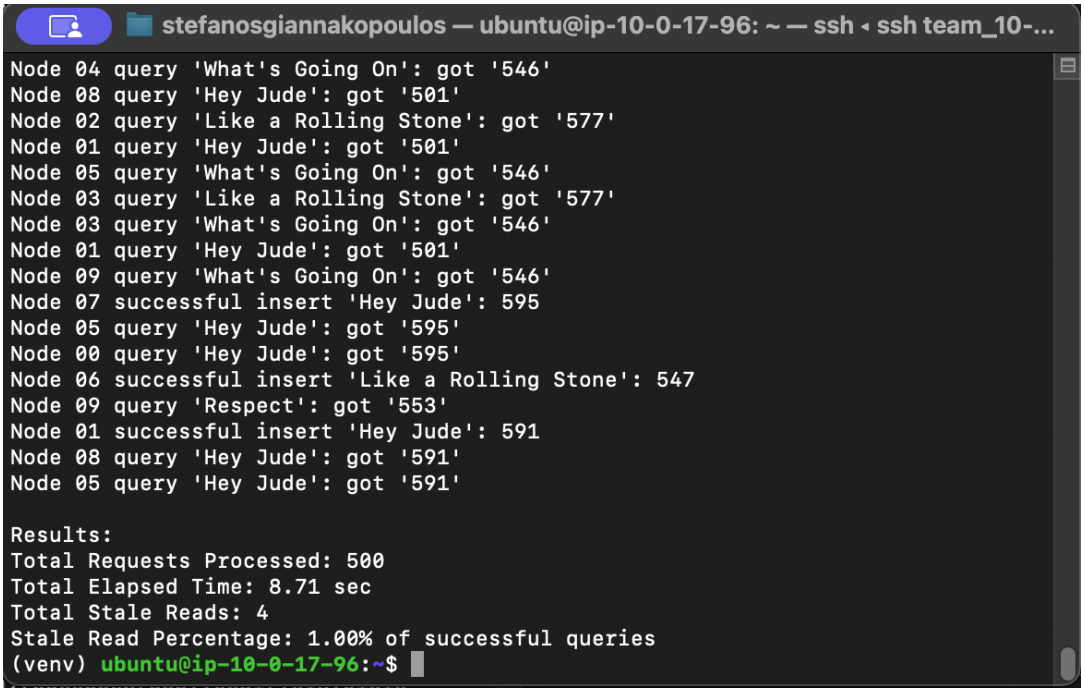


Παρατήρηση:

→ Δεν παρατηρήθηκαν stale reads. Η αυστηρή σειρά εγγραφών και αναζητήσεων εγγυάται φρέσκα δεδομένα.

2.Για eventual consistency και k=3:

Στο Eventual Consistency οι εγγραφές γίνονται άμεσα στον τοπικό κόμβο και το replication πραγματοποιείται ασύγχρονα. Οι queries μπορούν να επιστρέψουν παλιότερη τιμή αν δεν έχει ολοκληρωθεί το replication σε όλους τους κόμβους.



Παρατήρηση:

→ Παρατηρήθηκαν 4 stale reads (τιμές που δεν είναι fresh), και αυτό οφείλεται στο γεγονός ότι μέχρι να γίνει το replication στην k-αδα , μπορεί να γίνει query όπως και γίνεται και το αποτέλεσμα επιστρέφει από κόμβο που το replication δεν έχει πραγματοποιηθεί .

3.Συμπεράσματα

Η επιλογή consistency μοντέλου εξαρτάται από τις απαιτήσεις της εφαρμογής:

- Το **Chain Replication** προσφέρει αυστηρή συνέπεια (linearizability), ιδανικό για συστήματα όπου απαιτείται πάντα η πιο πρόσφατη τιμή, όπως χρηματοοικονομικά συστήματα ή συστήματα κράτησης.
- Το **Eventual Consistency** προσφέρει υψηλότερη διαθεσιμότητα και απόδοση, κατάλληλο για εφαρμογές όπου κάποιες καθυστερήσεις στην ενημέρωση των δεδομένων είναι αποδεκτές, όπως συστήματα κοινωνικών δικτύων.