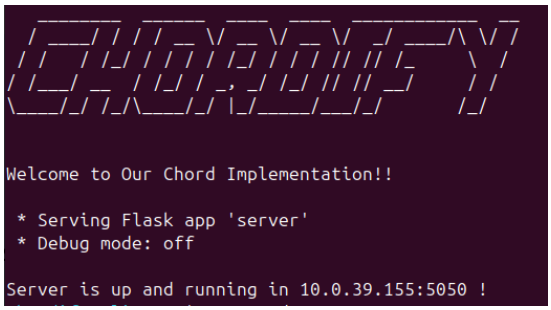
Distributed Systems 2025 - Chordify

Παναγιώτης Μπέλσης el20874
Θεοδοσίου Γεώργιος el20109
Μαρντιροσιάν Φίλιππος el20034

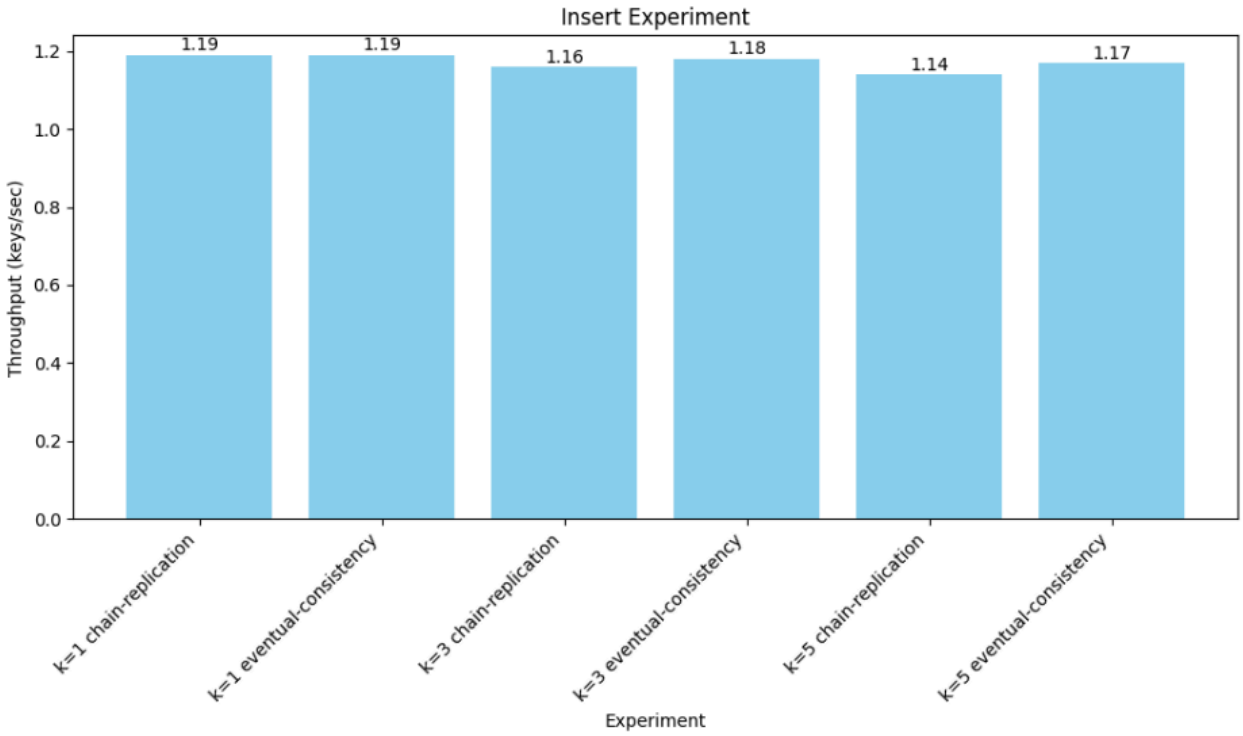


(!) Στην υλοποίησή μας, για την περίπτωση του linearizability, διαλέξαμε **chain replication**.

Αρχικά, για τα πειράματα, τρέξαμε τους 10 διαφορετικούς κόμβους, κι άρα 10 διαφορετικούς servers:

```
chordify-cli@NTUA$ overlay
Info of whole Cluster:
+-----------------------------------------------------+-------------+-----------+
|                        Hash                         |   Node IP   | Node Port |
+-----------------------------------------------------+-------------+-----------+
|  163162225042823161569705556514841128156452882 39  |  10.0.39.19 |    5050   |
|  276978885861263613669631433157681183875789884433 | 10.0.39.177 |    5051   |
|  331251593873405606612842059573183589924521314196 | 10.0.39.191 |    5051   |
|  385338712584612390161671209065390954204480091433 | 10.0.39.155 |    5050   |
|  410271491015581887573721664697587045373430772685 |  10.0.39.87 |    5051   |
|  464885683357206364287581611648946226323593727849 | 10.0.39.191 |    5050   |
|  704553053960351693130932751670118618832360342742 | 10.0.39.177 |    5050   |
|  886717963639729865265329820495952638984085957360 | 10.0.39.155 |    5051   |
|  948765744154304373642574448984047899915517574524 |  10.0.39.19 |    5051   |
| 1388561491943384733156711398784452934359287954795 |  10.0.39.87 |    5050   |
+-----------------------------------------------------+-------------+-----------+
```

Write Throughput:
Για την διερεύνηση του write throughput, κληθήκαμε να ελέγξουμε την συμπεριφορά του συστήματός μας απέναντι σε διαδοχικά inserts, όπως δίνονται στα αρχεία insert_00_part.txt,insert_01_part.txt …, για τους 6 διαφορετικούς συνδυασμούς replication factor/consistency type που υποδεικνύονται από την εκφώνηση



Στην περίπτωση του linearizability, αύξηση του k συνεπάγεται αύξηση του χρόνου ανά insertion. Αυτό συμβαίνει διότι σύμφωνα με το chain replication που έχει υλοποιηθεί, ένα write ολοκληρώνεται μόνο αφότου πραγματοποιηθεί εγγραφή και στους k replica managers, με τον τελευταίο να είναι υπεύθυνος για την επιστροφή του αποτελέσματος του write στον client.

Στην περίπτωση του eventual consistency το write ολοκληρώνεται αμέσως μετά την εγγραφή στον primary replica manager, ο οποίος απαντά σχετικά στον client. Άρα περιμένουμε παρόμοια αποτελέσματα όσο και αν είναι το k.

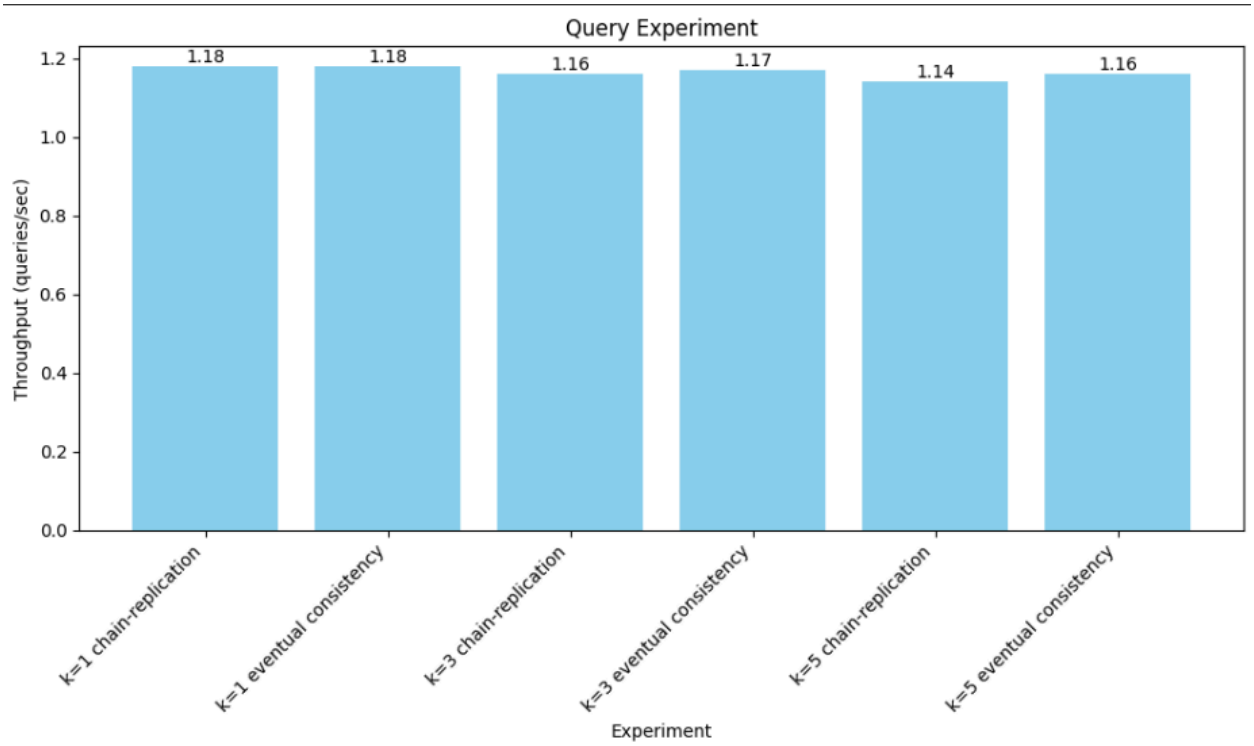
Για k=1 ο χρόνος για linearizability και eventual consistency είναι ο ίδιος, αφού και στις δύο περιπτώσεις αναζητείται ο υπεύθυνος κόμβος για το προς εισαγωγή κλειδί και το write ολοκληρώνεται με την εγγραφή σ' αυτόν (δεν υπάρχουν αντίγραφα), με αποτέλεσμα ο ίδιος να απαντά στον client σχετικά.

Όσο αυξάνεται, ωστόσο, το replication factor, η συμπεριφορά του συστήματος για linearizability αποκλίνει, αφού στην περίπτωση αυτή απαιτείται, μετά την εισαγωγή του κλειδιού στον πρωτεύοντα κόμβο, η διάδοση και αποθήκευση των αντιγράφων στους επόμενους k-1 replica managers, με τον τελευταίο εξ αυτών (replication tail) να είναι υπεύθυνος για την απάντηση στο χρήστη (ολοκλήρωση write).

Στο eventual consistency, δεδομένου του ότι, ανεξαρτήτως του replication factor, τα write επιστρέφουν (ολοκληρώνονται) μετά την εγγραφή στον primary replica manager, θα περιμέναμε το throughput να παραμένει σταθερό και στα 3 πειράματα.

Read Throughput:

Για τον έλεγχο του read throughput δοκιμάσαμε την ανταπόκριση των 6 περιπτώσεων replication factor / consistency, σε μια σειρά διαδοχικών queries, όπως αυτά βρίσκονται στα αρχεία query_00.txt, query_01.txt...



Στην περίπτωση του **eventual consistency**, είναι εμφανές πως ο χρόνος που απαιτείται για να ολοκληρωθεί ένα query, μειώνεται για μεγαλύτερες τιμές του k, δηλαδή για μεγαλύτερο αριθμό replicas. Το αποτέλεσμα αυτό είναι αναμενόμενο, καθώς μεγαλύτερος αριθμός replicas, εξασφαλίζει μεγαλύτερη πιθανότητα να βρεθεί αντίγραφο της ζητούμενης εγγραφής σε περιοχή κοντινή του κόμβου από τον οποίο ξεκινάει το request (ή στον κόμβο τον ίδιο), ενώ στη χειρότερη περίπτωση, κατά την οποία η ζητούμενη εγγραφή δεν υπάρχει στο σύστημα, αυτό θα διαπιστωθεί μόλις το query φτάσει στο υπεύθυνο βάσει ID κόμβο (primary).
Στο **linearizability**, αύξηση του replication factor συνεπάγεται αύξηση του απαιτούμενου χρόνου για την ολοκλήρωση του query. Στην περίπτωση αυτή, λόγω του chain replication, η ανάγνωση για κάθε κλειδί μπορεί να γίνει μόνο από έναν συγκεκριμένο κόμβο, το replication tail της ζητούμενης εγγραφής. Σύμφωνα με την παρατήρηση αυτή, ο χρόνος αναζήτησης στο δακτύλιο εξαρτάται άμεσα από τον αριθμό κόμβων N του συστήματος, αφού στη χειρότερη περίπτωση ένα query θα χρειαστεί να προωθηθεί σε ολόκληρη την αλυσίδα μέχρι να φτάσει στον υπεύθυνο για την απάντηση κόμβο (O(N)). Στη δική μας περίπτωση, ωστόσο, τα πειράματα πραγματοποιούνται για σταθερό αριθμό κόμβων (10).

> 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. Καταγράψτε τις απαντήσεις των queries σε περίπτωση linearization και eventual consistency. Ποια εκδοχή μας δίνει πιο fresh τιμές;

Αρχικά, αν τα scripts που εκτελούν τα requests τρέχουν παράλληλα, η σειρά με την οποία εκτελούνται τα αιτήματα μπορεί να διαφέρει από εκτέλεση σε εκτέλεση.

Παρόλα αυτά, οι απαντήσεις που λαμβάνονται από τα queries παρουσιάζουν διαφορές ανάμεσα στο linearizability και το eventual consistency. Το chain replication (που διασφαλίζει linearizability) επιστρέφει πιο πρόσφατες (fresh) τιμές, καθώς ακολουθεί αυστηρή σειρά ενημερώσεων. Αντίθετα, στο eventual consistency, όπου η ανάγνωση γίνεται από το πρώτο διαθέσιμο αντίγραφο που εντοπίζεται, υπάρχει η πιθανότητα να επιστραφεί μια μη ενημερωμένη (stale) τιμή, εφόσον το συγκεκριμένο αντίγραφο δεν έχει ακόμα συγχρονιστεί με τις πιο πρόσφατες αλλαγές.

Στα πειράματα που πραγματοποιήθηκαν, όταν το σύστημα αποτελείται από 10 κόμβους, δεν παρατηρήθηκε αξιοσημείωτη διαφορά στις τιμές που επιστρέφουν τα queries. Ωστόσο, με μόλις 3 κόμβους, η διαφορά γίνεται εμφανής, με το chain replication να εξασφαλίζει πιο πρόσφατες και ακριβείς τιμές, ενώ το eventual consistency ενδέχεται να επιστρέψει παλαιότερες καταχωρήσεις λόγω της χαλαρότερης πολιτικής συγχρονισμού του.