ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ ΣΤΑ ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Ακ. έτος 2024-2025, 9ο Εξάμηνο, Σχολή ΗΜ&ΜΥ

Τελική Ημερομηνία Παράδοσης: **3 εβδομάδες μετά το τέλος της εξεταστικής**

Περιγραφή

Στην εργασία αυτή θα σχεδιάσετε και θα υλοποιήσετε το Chordify, μια P2P εφαρμογή ανταλλαγής τραγουδιών, βασισμένη στο Chord DHT[1]. Για να χρησιμοποιήσουν την εφαρμογή οι χρήστες συνδέονται σε ένα Chord DHT και μοιράζονται με τους υπόλοιπους τα τραγούδια που έχουν. Μπορούν να κάνουν επίσης αναζήτηση τραγουδιού με βάση τον τίτλο του, ώστε να βρουν ποιος/ποιοι κόμβοι το έχουν. Στην υλοποίηση δεν θα περιλαμβάνεται η μεταφόρτωση του ίδιου του τραγουδιού. Δεν είναι απαραίτητο να υλοποιήσετε τα finger tables ούτε τη δρομολόγηση μέσω αυτών (αν και μπορεί να σας δώσει bonus πόντους). Επίσης δε θα χρειαστεί να χειριστείτε αποτυχίες κόμβων. Οι βασικές λειτουργίες που θα υλοποιήσετε θα είναι (a) η διαίρεση του χώρου των Ids (κόμβοι και αντικείμενα), (b) δρομολόγηση δακτυλίου, (c) η εισαγωγή κόμβων, (d) η αναχώρηση κόμβων και (e) το replication των δεδομένων.

Η εφαρμογή που θα αναπτύξετε θα είναι ένα file sharing application με πολλαπλούς κατανεμημένους κόμβους DHT. Κάθε κόμβος θα πρέπει να υλοποιεί όλες τις λειτουργίες του DHT, για παράδειγμα να δημιουργεί server και client processes, να ανοίγει sockets, να απαντά σε εισερχόμενα αιτήματα. Επίσης θα πρέπει να υλοποιεί μια απλοποιημένη εκδοχή του πρωτοκόλλου δρομολόγησης του Chord. Τέλος, θα πρέπει να χειρίζεται την εισαγωγή και αποχώρηση (graceful departure) κόμβων. Οι απαιτήσεις για τον κάθε κόμβο είναι οι εξής:

1. Ο κόμβος πρέπει να υλοποιεί όλες τις λειτουργίες για την επικοινωνία με τους υπόλοιπους κόμβους (server/client threads/processes, sockets)
2. Κάθε κόμβος λαμβάνει μοναδικό id που προέρχεται από την εφαρμογή μιας hash function[1] στον συνδυασμό ip_address:port, όπου ip_address η ip του κόμβου και port η πόρτα στην οποία ακούει για αιτήματα η εφαρμογή.
3. Κάθε κόμβος υλοποιεί τις λειτουργίες insert(key, value), query(key) και delete(key) για <key, value> ζεύγη όπου τόσο το key όσο και το value είναι strings. Θυμηθείτε ότι το key πρέπει να περάσει από hash function πριν χρησιμοποιηθεί για οποιαδήποτε από τις παραπάνω λειτουργίες (ώστε να βρεθεί η σωστή θέση στον δακτύλιο). Για το Chordify, το key είναι το όνομα του τραγουδιού και value είναι η ip του κόμβου όπου υπάρχει το τραγούδι. Στη συγκεκριμένη εργασία δε θα ασχοληθούμε με το κομμάτι του data transfer, παρά μόνο με τον εντοπισμό του τραγουδιού, οπότε το value μπορείτε να θεωρήσετε ότι είναι ένα τυχαίο string.
4. Όταν η insert(key, value) καλείται με key που υπάρχει ήδη αποθηκευμένο στο DHT λειτουργεί ως update, δηλαδή προσθέτει στην παλιά τιμή του value την καινούρια (concat).
5. Για τη λειτουργία query(key) πρέπει να ληφθεί μέριμνα για τον ειδικό χαρακτήρα "*", που θα πρέπει να επιστρέφει όλα τα <key, value> ζεύγη που είναι αποθηκευμένα σε ολόκληρο το DHT ανά κόμβο.

[1] Για hash function χρησιμοποιήστε την SHA1. Η υλοποίησή της υπάρχει έτοιμη σε πολλές βιβλιοθήκες.

6. Ο κόμβος θα υλοποιεί δρομολόγηση δακτυλίου, ακολουθώντας τον σχεδιασμό του Chord. Αυτό σημαίνει ότι ο κόμβος θα κρατά δείκτες στον προηγούμενο και στον επόμενο κόμβο του λογικού δακτυλίου (ή σε λογαριθμικό αριθμό κόμβων αν θέλετε να υλοποιήσετε finger tables) και θα προωθεί οποιοδήποτε αίτημα στον επόμενό του, μέχρι το αίτημα να φτάσει στον σωστό κόμβο. Μόλις ο σωστός κόμβος παραλάβει το αίτημα το επεξεργάζεται και απαντά απευθείας στον κόμβο που ξεκίνησε το αίτημα. Επαναλαμβάνεται ότι δεν είναι απαραίτητο να υλοποιήσετε finger tables ούτε δρομολόγηση με χρήση τους, ωστόσο είστε ευπρόσδεκτοι να προσπαθήσετε εάν το θελήσετε
7. Το σύστημά σας θα πρέπει να χειρίζεται εισαγωγές νέων κόμβων (join(nodeID)) και αποχωρήσεις κόμβων (depart(nodeID)). Για τον σκοπό αυτό θα πρέπει να ορίσετε έναν κόμβο που θα δέχεται όλα τα αιτήματα για join (bootstrap κόμβος). Ο κόμβος αυτός θεωρούμε ότι είναι σταθερά συνδεδεμένος (δεν αποσυνδέεται ποτέ) και η ip του είναι γνωστή σε όλους. Προφανώς είναι και ο πρώτος κόμβος που εισέρχεται στο σύστημα. Κατά την εισαγωγή αποχώρηση ενός κόμβου θα πρέπει οι κόμβοι που επηρεάζονται να ενημερώσουν σωστά τους δείκτες στον προηγούμενο και επόμενο κόμβο που είναι απαραίτητοι για τη δρομολόγηση μηνυμάτων και να ανακατανείμουν τα κλειδιά τους ώστε ο κάθε κόμβος να είναι υπεύθυνος για τα σωστά κλειδιά. Δε χρειάζεται να ασχοληθείτε με την περίπτωση ταυτόχρονων join/depart - Θεωρείστε ότι ένας κόμβος εισάγεται ή αποχωρεί αφού ολοκληρωθεί η εισαγωγή ή αποχώρηση του προηγούμενου. Δε χρειάζεται να χειριστείτε αιτήματα insert/delete/query ταυτόχρονα με τα join/depart. Θεωρείστε ότι αυτά τα αιτήματα έρχονται αφού το σύστημα έρθει σε ισορροπία. Δε χρειάζεται να ασχοληθείτε με αποτυχίες κόμβων. Θεωρούμε ότι οι κόμβοι φεύγουν από το σύστημα μόνο οικειοθελώς μέσω της διαδικασίας depart.
8. Αφού ελέγξετε ότι δουλεύουν οι βασικές λειτουργίες που περιγράφονται παραπάνω, θα εισάγετε και replication των <key, value> δεδομένων που είναι αποθηκευμένα στο σύστημα. Το replication factor θα είναι μια μεταβλητή k (θα γίνουν μετρήσεις με διαφορετικές τιμές του k). Αυτό σημαίνει ότι κάθε <key,value> ζεύγος θα πρέπει να αποθηκεύεται εκτός από τον κόμβο που είναι υπεύθυνος για το hash(key) και στους k-1 επόμενους κόμβους στον λογικό δακτύλιο. Το replication θα πρέπει να ληφθεί υπόψιν σε όλες τις βασικές λειτουργίες του DHT (insert, delete, query, join, depart).
9. Θα υλοποιήσετε 2 είδη συνέπειας (consistency) για τα replicas: (a) linearizability και (b) eventual consistency.

(a) Για την περίπτωση του **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. Για linearizability μπορείτε να υλοποιήσετε είτε quorum replication είτε chain replication.

Quorum replication: Σε αυτήν την περίπτωση, για να επιτύχουμε linearizability θα πρέπει και το read quorum και το write quorum να είναι ίσο με k (γιατί;). Ένας κόμβος θα είναι ο coordinator (ποιος;). Ο coordinator για οποιοδήποτε read/write θα πρέπει πάντα να επικοινωνεί με τους υπόλοιπους k-1 κόμβους που έχουν replicas για να διαβάσει ή να γράψει μια τιμή. Για write operation, όλα τα values μπορούν να έχουν versions για να ξεχωρίζουμε παλιά από πρόσφατα αντίγραφα. Για read operations αν οι κόμβοι στο read quorum έχουν διαφορετικές versions του ίδιου αντικειμένου, επιστρέφεται το πιο πρόσφατο αντίγραφο.

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.

(b) Για την περίπτωση του **eventual consistency** οι αλλαγές θα 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).

Chordify client

Θα πρέπει να υλοποιήσετε έναν client (ένα απλό cli αρκεί, αλλά αν θέλετε να είστε πιο fancy θα μετρήσει θετικά) που θα δίνει στον χρήστη τη δυνατότητα να εκτελεί τα παρακάτω:

insert <key> <value>

Εισαγωγή δεδομένων: Γίνεται εισαγωγή ενός νεου (key,value) pair, όπου key είναι ο τίτλος του τραγουδιού και value ένα οποιοδήποτε string (που υποτίθεται ότι δίνει τον κόμβο που πρέπει να συνδεθούμε για να κατεβάσουμε το τραγούδι). Θα καλεί αντίστοιχη συνάρτηση στο backend που θα υλοποιεί την παραπάνω λειτουργία.

delete <key>

Διαγραφή δεδομένων: Γίνεται η διαγραφή του (key, value) pair με κλειδί key. Θα καλεί αντίστοιχη συνάρτηση στο backend που θα υλοποιεί την παραπάνω λειτουργία.

query <key>

Αναζήτηση δεδομένων: Αναζητείται το key και επιστρέφεται το αντίστοιχο value του από τον κόμβο που είναι υπεύθυνος για το key αυτό (ή από κάποιον replica κόμβο). Στην ειδική περίπτωση που δοθεί ως key το * θα επιστρέφονται όλα τα <key, value> ζεύγη που είναι αποθηκευμένα σε ολόκληρο το DHT ανά κόμβο. Καλεί αντίστοιχη συνάρτηση στο backend που υλοποιεί την παραπάνω λειτουργία.

depart

Αποχώρηση κόμβου: Ο κόμβος αποχωρεί gracefully από το σύστημα. Καλεί αντίστοιχη συνάρτηση στο backend που υλοποιεί την παραπάνω λειτουργία.

overlay

Εκτύπωση της τοπολογίας του δικτύου: Εκτυπώνονται οι κόμβοι στο δακτύλιο του chord με τη σειρά με την οποία είναι συνδεδεμένοι. Καλεί αντίστοιχη συνάρτηση στο backend που υλοποιεί την παραπάνω λειτουργία.

help

Επεξήγηση των παραπάνω εντολών.

Πειράματα

Θα αναπτύξετε το Chordify σε όποια γλώσσα προγραμματισμού θέλετε. Θα το στήσετε σε υποδομή που θα σας παραχωρηθεί από την Amazon (θα σας δοθούν ακριβείς οδηγίες). Για την αναφορά θα εκτελέσετε τα παρακάτω πειράματα:

Για την αναφορά θα εκτελέσετε τα παρακάτω πειράματα:

- 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; Γιατί;

- 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. Τι γίνεται όσο το k αυξάνεται; Γιατί;
- 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;

[1] Stoica, Ion, et al. "Chord: A scalable peer-to-peer lookup service for internet applications." ACM SIGCOMM Computer Communication Review 31.4 (2001): 149-160.

Παραδοτέο της άσκησης θα είναι ο πηγαίος κώδικας (tarball με τα σχετικά αρχεία) καθώς και ένα pdf που θα παρουσιάζει τα αποτελέσματα των πειραμάτων. Επίδειξη της άσκησης θα γίνει σε συνεννόηση με τους διδάσκοντες μετά τη λήξη της προθεσμίας για το παραδοτέο.

Στο ηλεκτρονικό κείμενο να αναφέρετε στην αρχή τα στοιχεία σας (Όνομα, Επώνυμο, ΑΜ).

Ο κώδικας και η αναφορά θα παραδοθούν ηλεκτρονικά στην ιστοσελίδα:

http://www.cslab.ece.ntua.gr/courses/distrib/submit

Δουλέψτε σε ομάδες 2-3 ατόμων. Έχει ιδιαίτερη αξία για την κατανόηση του μαθήματος να κάνετε μόνοι σας την εργασία. Μην προσπαθήσετε να την αντιγράψετε από άλλους συμφοιτητές σας.