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 2016-2017, 9ο Εξάμηνο, Σχολή ΗΜ&ΜΥ
Τελική Ημερομηνία Παράδοσης: 10/03/2017

Περιγραφή

Στην εργασία αυτή θα σχεδιάσετε ένα απλό DHT, απλοποιημένη έκδοση του Chord [1]. Δεν είναι απαραίτητο να υλοποιήσετε τα finger tables ούτε τη δρομολόγηση μέσω αυτών. 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 ένας emulator που θα σηκώνει πολλαπλούς κόμβους του 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 αύξοντα αριθμό της. Π.χ. ο πρώτος κόμβος θα έχει id = hash("1"), ο δεύτερος hash("2") κοκ.
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 stings. 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)
4. Όταν η insert(key, value) καλείται με key που υπάρχει ήδη αποθηκευμένο στο DHT λειτουργεί ως update, δηλαδή σβήνει την παλιά τιμή του key και αποθηκεύει την καινούρια.
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.
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

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

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 (προφανώς τον αρχικό σας κόμβο δηλ. τον "1") που θα δέχεται όλα τα αιτήματα για join/depart. 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 reader quorum και το writer quorum να είναι ίσο με k-1 (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 reader 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).

Πειράματα

Για την αναφορά θα εκτελέσετε τα παρακάτω πειράματα:

- Εισάγετε 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 (δλδ 6 πειράματα) και καταγράψτε το write throughput του συστήματος (Πόσο χρόνο πήρε η εισαγωγή των κλειδιών προς τον αριθμό τους). Τα inserts θα ξεκινούν κάθε φορά από τυχαίο κόμβο 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 στο αρχείο query.txt και καταγράψτε 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. Τι γίνεται όσο το k αυξάνεται; Γιατί;
- Για DHT με 10 κόμβους και k=3, εκτελέστε 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. 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 ηλεκτρονικό κείμενο (**pdf**, **docx** ή **odt**) 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.