

Εξαμηνιαία εργασία στα Κατανεμημένα Συστήματα

Chordify

Σπυρίδων Γαλανόπουλος (03120063) - Πλάτανος Χαράλαμπος (03120108)
Ανδρέας Στάμος (03120018)

1 Εισαγωγή

Στην παρούσα εξαμηνιαία εργασία υλοποιήσαμε μια απλουστευμένη εκδοχή του Chord DHT, το Chordify σε Python. Ειδικότερα, δόθηκε έμφαση στην υλοποίηση της δομής του δακτυλίου και της δρομολόγησης σε αυτόν, στην εισαγωγή και αναχώρηση κόμβων και τέλος στο replication των δεδομένων (επιλέξαμε chain replication). Ακόμη, υποθέσαμε ότι δεν υπάρχουν αποτυχίες κόμβων καθώς και ότι τα όποια αιτήματα στο σύστημα έρχονται αφού αυτό σύστημα έρθει σε ισορροπία. Επιπροσθέτως, υλοποιήσαμε και δρομολόγηση μέσω finger tables (λογαριθμικού μεγέθους ως προς το πλήθος των κόμβων του συστήματος). Τέλος, φτιάξαμε και ένα web app για πιο άμεση και εύκολη αλληλεπίδραση με το σύστημα.

2 Πειράματα

Με το script cli/benchmark.py αυτοματοποιήσαμε την εκτέλεση των πειραμάτων και για τις τρεις περιπτώσεις. Συγκεκριμένα, τρέξαμε τα insert, query και request πειράματα και για τους δύο τύπους consistency για αριθμό αντιγράφων 1, 3, και 5. Τα αποτελέσματα και ο σχολιασμός παρουσιάζονται ακολούθως.

2.1 Insertions

Από το διάγραμμα παρατηρούμε ότι το write throughput και στις δύο περιπτώσεις consistency μειώνεται με την αύξηση replication factor (φυσικά για την περίπτωση $k = 1$ όπου δεν έχουμε replication πρακτικά έχουμε το ίδιο write throughput περίπου). Ειδικότερα, στην περίπτωση του linearizability η μείωση αυτή γίνεται με πιο απότομο ρυθμό από την περίπτωση του eventual consistency. Η μείωση που παρατηρούμε είναι λογική καθώς με την αύξηση του k κάθε κόμβος είναι υπεύθυνος για την διατήρηση περισσότερων αρχείων με αποτέλεσμα να πρέπει να γίνουν περισσότερα writes και ανταλλαγές μηνυμάτων μεταξύ των κόμβων σε κάθε insert αυξάνοντας τον χρόνο. Μάλιστα, για το linearizability επειδή τα writes πρέπει να γίνουν propagate στους επόμενους $k - 1$ κόμβους που έχουν το αντίγραφο κατευθείαν η μείωση στο throughput είναι ακόμα πιο έντονη, σε αντίθεση με το eventual consistency όπου αυτό γίνεται lazily (αλλά και πάλι γίνονται παραπάνω HTTP requests οπότε έχουμε μείωση του throughput).

2.2 Queries

Από το διάγραμμα παρατηρούμε ότι με την αύξηση του replication factor το read throughput στην περίπτωση του linearizability μειώνεται ενώ στην περίπτωση του eventual consistency αυξάνεται (φυσικά για την περίπτωση $k = 1$ όπου δεν έχουμε replication πρακτικά έχουμε το ίδιο read throughput περίπου). Πράγματι, για το linearizability, με την αύξηση του k αφού θα πρέπει σε κάθε read να επιστρέφουμε την τιμή του τελευταίου κόμβου στο chain οπότε θα πρέπει να διατρέχουμε όλο και περισσότερους κόμβους για να φτάσουμε σε αυτόν. Αντίθετα, στην περίπτωση του eventual consistency, η ανάγνωση μπορεί να γίνει από οποιοδήποτε αντίγραφο συνεπώς με την αύξηση του k η πιθανότητα ένας κόμβος να έχει το ζητούμενο αρχείο (αντίγραφο) είναι μεγαλύτερη με αποτέλεσμα να χρειάζονται on expectation λιγότερα βήματα για την εύρεση του αρχείου, με αποτέλεσμα να παρατηρούμε αύξηση του read throughput με την αύξηση του replication factor.

2.3 Requests