Διαχείρηση Μεγάλων Δεδομένων

Γεώργιος Παπαποστόλου
A.M.: 7115132200005

Για την υλοποίηση της εργασίας χρησιμοποίησα τη γλώσσα Java (έκδοση jdk-11.0.16.1). Μέσα στο αρχείο .zip θα βρείτε τρεις φακέλους, έναν για το κάθε κομμάτι της εργασίας. Τα αρχεία Java θα χρειαστεί να τα κάνετε compile (javac *.java) και για να τα τρέξετε μπορείτε να χρησιμοποιήσετε την εντολή java + την εντολή που αναγράφετε στην εκφώνηση της εργασίας (π.χ. java genData keyFile.txt 1000 3 4 5).

Για τους ελέχους που έχω κάνει χρησιμοποιούσα την IP 127.0.0.1 για να συνδεθούν οι servers. Στους ελέγχους αυτούς δοκίμασα να τρέχει μόνο ένας client και δέκα servers , και τα κλειδιά που είχα παράξει και πέρασα στους servers ήταν 150.000.

1 Assumptions

Οι υποθέσεις που έκανα κατά την υλοποίηση της εργασίας είναι οι ακόλουθες:

- Όλες οι μεταβλητές string χρείαζεται να περιβάλλονται από " (π.χ. "key0").
- Τα float και int των queries που κατασκευάζω παίρνουν τιμές από 0 έως 1000.
- Τα βελάκια που υπάρχουν στα ζευγάρια key-value χρειάζεται να έχουν κενά αριστερά και δεξιά τους (->) ομοίως και για το σύμβολο που διαχωρίζει τα διαδοχικά υπο-κλειδιά (|).
- Κατά την εκκίνηση του client όλοι οι servers που αναγράφονται στο αρχείο πρέπει να τρέχουν.
- Οι μεταβλήτες που δίνονται στην εντολή COMPUTE είναι μήκους 1.
- Η εντολή COMPUTE υπολογίζεται σε κάθε server.

2 Data creation

Για τη δημιουργία των δεδομένων έφτιαξα μία κλάση genData, η οποία περιέχει τη main συνάρτηση, και μία συνάρτηση FillQuery η οποία κατασκευάζει τα queries αναδρομικά. Αρχικά στη main ελέγχω αν οι παράμετροι είναι σωστού τύπου, και ανάλογα με τη τιμή τους ξεκινάει η δημιουργία των queries. Αν το m έιναι 0, τότε θα επιστρέψω ένα κλειδί με empty value (δηλαδή "key0" -> []), διαφορετικά καλώ τη συνάρτηση FillQuery. Παράλληλα για την ευκολότερη διαχείριση των δεδομένων, έφτιαξα μία βοηθητική κλάση KeyValueData την οποία χρησιμοποίησα για να αποθηκεύσω τα κλειδιά και μία λίστα με τις τιμές που είναι τύπου key-value.

Η συνάρτηση FillQuery δέχεται ως παραμέτρους ένα αντικείμενο KeyValueData που θα αποτελέσει τον "πατέρα" των κλειδιών που θα παραχθούν, το βάθος του κλειδιού, τον αριθμό των κλειδιών που μπορεί να έχει ως value και τέλος το query όπως έχει φτιαχτεί μέχρι στιγμής. Αρχικά η FillQuery διαλέγει ποιο value της θα έχει key-value ζευγάρι ως τιμή (kVChild μεταβλητή) και στη συνέχεια ξεκινάει μία επανάληψη μέχρι να φτάσουμε τον επιθυμητό αριθμό κλειδιών. Μέσα στην επανάληψη επιλέγεται μία τυχαία γραμμή από το αρχείο keyFile.txt ώστε να βρεθεί το όνομα του κλειδιού και ο τύπος του value του. Εφόσον το depth είναι θετικό θα κληθεί ξανά η FillQuery. Η αναδρομή θα τελείωσει όταν η FillQuery κληθεί με depth = 0. Όλα τα υπόλοιπα values θα συμπληρωθούν από τη συνάρτηση RandomValue.

Η συνάρτηση RandomValue φτιάχνει ένα τυχαίο value ανάλογα με τον τύπο που έχει επιλεχθεί από το αρχείο (τα int και τα float είναι τυχαίοι αριθμοί από το 0 έως το 1000) ή μία άδεια τιμή (δηλαδή []).

Τέλος έχω υποθέσει ότι όλα τα εξωτερικά κλειδιά είναι της μορφής "key0", "key1", ..., καθώς και ότι όλα τα strings πρέπει να περιέχονται σε εισαγωγικά (" "). Το path του αρχείο που αποθηκεύονται τα queries έχει ορισθεί εσωτερικά (στη συνάρτηση main) να είναι το directory των αρχείων.

3 Key-Value client

Για την υλοποίηση του Key-value client έφτιαξα μία κλάση KVClient η οποία περιέχει τις συναρτήσεις main, ClientReceiveMessage και ClientSendMessage. Στη main ο Client συνδέεται στους server με βάση τις IP και τα ports που διάβασε από το αρχείο και αποθηκεύει τα sockets σε μία λίστα. Στη συνέχεια καλεί τη συνάρτηση ClientReceiveMessage για κάθε socket και τη συνάρτηση ClientSendMessage με τη λίστα όλων των sockets.

Η συνάρτηση ClientReceiveMessage τρέχει για κάθε socket σε διαφορετικό Thread και περιμένει να τυπώσει το μήνυμα που δέχεται από κάθε server. Παράλληλα αν χαθεί η σύνδεση με κάποιον server αφαιρείται από τη λίστα.

Η συνάρτηση ClientSendMessage αρχικά στέλνει τα queries που της έχουν δωθεί σε k servers. Για να επιτύχω το k replication στέλνω το μήνυμα στους πρώτους k servers και στη συνέχεια ανακατέβω τη λίστα των servers . Τέλος πριν στείλω οποιοδήποτε query ελέγχω τον αριθμό των servers στους οποίους ο client είναι συνδεδεμένος και τυπώνω το κατάλληλο μήνυμα.

4 Key-Value server

Για την υλοποίηση του Key-value server έφτιαξα μία κλάση KVServer η οποία περιέχει τις συναρτήσεις main και ServerClientCom. Στη main διαβάζονται μόνο τα inputs και στη συνέχεια καλείται η συνάρτηση ServerClientCom.

Η συνάρτηση ServerClientCom περιμένει να συνδεθεί κάποιος client και αφού συνδεθεί περιμένει να λάβει κάποιο μήνυμα, και ανάλογμα το μήνυμα θα επιστρέψει την κατάλληλη απάντηση στον client.

Για την αποθήκευση, εύρεση και διαγραφή των εξωτερικών κλειδιών έχω υλοποιήσει μία δομή Trie, ενώ για τα εσωτερικά κλειδιά έχω χρησιμοποιήσει τη κλάση KeyValueNode, όμοια με αυτή στην δημιουργία των queries. Κάθε KeyValueNode περιέχει το string του κλειδιού, μία λίστα με τα values της που είναι KeyValueNodes και μία τιμή για το value της που είναι int, string ή float. Από την κατασκευή της δομής είτε η λίστα είτε η μεμονομένη τιμή θα είναι null.

Η αποθήκευση των εξωτερικών κλειδιών στο Trie έγινε με βάση τη τιμή της μεταβλητής key του KeyValueNode. Επίσης η συνάρτηση Find επιστρέφει ένα KeyValueNode με τιμή key το κλειδί που ζητήθηκε.

Όλες οι συναρτήσεις που υπολογίζουν τις εντολές PUT, GET, QUERY, DELETE και COMPUTE βρίσκονται στη κλάση QueryCalculations. Η εντολή PUT αρχικά χωρίζει το query σε δύο strings αριστερά και δεξιά από το " -> " ελέγχει αν ο αριθμός των αριστερών και δεξιών αγκυλών είναι ο ίδιος και στη συνέχεια αν το μήκος του αριστερού string είναι μεγαλύτερο του 2, δηλαδή δεν είναι μόνο οι αγκύλες, καλεί τη συνάρτηση SplitQuery η οποία αναδρομικά χωρίζει τα κλειδιά του query σε KeyValueNodes .

Για την εντολή GET βρίσκω το KeyValueNode μέσω της συνάρτησης Find του Trie και καλώ τη συνάρτηση BuildString η οποία προσπελάζει τα KeyValueNodes για να φτιάξει το string που πρέπει να επιστραφεί.

Για την εντολή QUERY αρχικά βρίσκουμε το εξωτερικό κλειδί και τα "υποκλειδιά" του. Αν δεν υπάρχουν "υποκλειδιά" τότε καλώ τη συνάρτηση GET αλλιώς για τα "υποκλειδιά" καλώ τη συνάρτηση SearchSubKeys η οποία ψάχνει να βρεί KeyValueNode με κλειδί το πρώτο από αυτά, μετά περνά στο επόμενο και επαναλαμβάνει

τη διαδικασία μέχρι να τα βρει όλα ή να προκύψει κάποιο σφάλμα.

Τέλος για την εντολή COMPUTE χώρισα τη μαθηματική έκφραση από τις μεταβλητές. Για τις μεταβλητές χρησιμοποιήσα τις συναρτήσεις που υπάρχουν από την εντολή QUERY βρήκα την τιμή τους και αν αντιστοιχούν σε αριθμό τις αντικαθιστώ στη μαθηματική έκφραση. Στη συνέχεια εφάρμοσα τον αλγόριθμο Shunting yard του Dijkstra, για να βρώ τη μαθηματική έκφραση σε Reverse Polish Notation και υπολόγισα την τελική της τιμή. Οι συναρτήσεις με την εφαρμογή του αλγόριθμου του Dijkstra και τον υπολογισμό της έκφρασης βρίσκονται στη κλάση ComputeMathExpression.