Αναστάσιος Μελιδώνης – 1115201700084

Αυτή η εργασία βασίζεται και χρησιμοποιεί τις δύο προηγούμενες που υλοποιήθηκαν. Η πρώτη εργασία δεν έχει αγγιχτεί καθόλου και η δεύτερη σχεδόν καθόλου πέρα από κάποια γενικά πράγματα προσαρμοσμένα στα δεδομένα της παρούσας εργασίας (π.χ, αλλαγή του WRITEfd και READfd στον worker έτσι ώστε να επικοινωνεί πλέον με τον server αντί για τον master).

Έχει υλοποιηθεί πλήρως multithreaded server καθώς και multithreaded client όπου επικοινωνούν μεταξύ τους όπως ακριβώς αναφέρεται στην εκφώνηση της εργασίας. Χρησιμοποιήθηκε TCP σύνδεση, καθώς είναι πιο αξιόπιστη σε σχέση με UDP σύνδεση, αλλά κυρίως επειδή λειτουργεί “out of the box” το πρωτόκολλο επικοινωνίας που είχαμε για τα named pipes από την δεύτερη εργασία.

Το προτόκολλο επικοινωνίας βελτιώθηκε ελφρώς ώστε να είμαστε σίγουροι ότι λειτουργεί αλάνθαστα στα sockets.

Ενώ στην αρχή η επικοινωνία workers - master γίνεται με bufferSize που δίνεται από τον χρήστη. Η επικοινωνία worker - server και αντιστοιχα server – client γίνεται με fixed bufferSize, το οποίο επιλέχθηκε μετά από αρκετό testing. Συγκεκριμένα με bufferSize = 128. Αυτή η επιλογή έγινε για λόγους ταχύτητας και σταθερότητας, καθώς έτσι όπως έχει υλοποιηθεί το προτόκολλο επικοινωνίας, ευνοείται από μικρό bufferSize.

Στον client και στον server οι μεταβλητές τις οποίες τα νήματα μπορούν να πειράξουν/επεξεργαστούν **προστατεύονται** με **mutexes** και **condition variables** ώστε να επιτευχθεί συγχρονισμός μεταξύ των threads, δίχως busy-waiting.

Γίνεται σωστό κλείσιμο μιας TCP σύνδεσης! Αυτό είναι πάρα πολύ σημαντικό, καθώς σε περίπτωση που ο σέρβερ μας λάβει πολλά αιτήματα θα κολλήσει και θα αποτύχει να κάνει establish συνδέσεις, επειδή με λανθασμένο κλείσιμο σύνδεσης μπαίνουν οι υπάρχουσες συνδέσεις σε CLOSE_WAIT state και δεν μπορει ο σερβερ να επαναχρησιμοποιήσει ports.

Έχει λυθεί το πρόβλημα του interleaving της εκτύπωσης αποτελεσμάτων. Αυτό έχει επιτευχθεί με το “χτίσιμο” της απάντησης σε μια μεταβλητή και με τη χρήση mutex κατά την εκτύπωση του αποτελέσματος.

Υπάρχει Handling μιας νέας σύνδεσης Client. Συγκεκριμένα υπάρχει διαχωρισμός μιας νέας σύνδεσης ενός Client και ενός Query. Αυτό επιτυγχάνεται απλά με την αποστολή ενός κενού μηνύματος. Ο server απλά αποδέχεται connections και διαβάζει/γράφει σε αυτά. Αν το πρώτο μήνυμα που θα λάβει από ένα connection είναι το κενό μήνυμα τότε σημαίνει καινούργιος client.

Έχει ελεγθεί διεξοδικά το πρόγραμμα για memory leaks! Έχει γίνει έλεγχος με τη χρήση valgrind σε whoServer, whoClient και master(με –trace-children=yes) και έχουν απαλοιφεί όλα τα leaks.

Όλα τα παραπάνω κάνουν τον σέρβερ να μπορεί να εξυπηρετήσει τεράστιο πλήθος από αιτήματα, να διαχειρίζεται clients και να τρέχει για μεγάλο χρονικό διάστημα.

Οδηγίες για compilation και execution

Για να κάνουμε compile το πρόγραμμά μας αρκεί να κάνουμε **make**

Θα πρέπει να τρέξουν τα προγράμματα με την εξής σειρά:

- `./whoServer –q (queryPortNum) -s (statisticsPortNum) –w (numThreads) –b (bufferSize)`
- `./master –w (numWorkers) -b (bufferSize) –s (serverIP) –p (serverPort) -i (input_dir)`
- `./whoClient –q (queryFile) -w (numThreads) –sp (servPort) –sip (servIP)`

Όπου (…) ορίζεται από τον χρήστη η τιμή.

Για έξοδο από το πρόγραμμα μας

- whoServer: CTRL + C.
- whoClient: κάνει μόνο του exit μόλις ολοκληρωθούν όλα τα queries.
- master: αρκεί να γράψουμε /**exit** στο τερματικό όπου κλείνουν master και workers.