

**Π**

**ΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

ΕΠΛ 371

# ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΥΣΤΗΜΑΤΩΝ

***Αριθμός Εργασίας:*** *4η*

***Τίτλος:*** *Ανάπτυξη πολυνηματικού Εξυπηρετητή Ιστού για πρωτόκολλο HTTP/1.1*

***Ονόματα:***  *Ανδρέας Κωστή, Παναγιώτης Χαραλάμπους, Στέφανη Νικολάου*

***Α.Φ.Τ:*** *1003060, 952304, 932787*

***Ημερομηνία Παράδοσης:*** *12/4/17*

**Αρχιτεκτονική Συστήματος**

Με σκοπό να υλοποιήσουμε ένα εξυπηρετητή που θα μπορεί να εξυπηρετεί πολλούς χρήστες ταυτόχρονα σε διάφορους ιστούς χρησιμοποιήσαμε νήματα. Με την υλοποίηση του σύστημά μας δημιουργούμε threadpool.Το μέγεθος του threadpool υπάρχει στο αρχείο του κώδικα που υλοποιήσαμε και με τον συγχρονισμό των νημάτων το πραγματοποιήσαμε επιτυχώς. Αν ξεπεράσουμε των αριθμό των διαθέσιμών νημάτων τότε μπαίνουν σε ένα queue έτσι ώστε όταν ένα νήμα θα γίνει ξανά διαθέσιμο να μπορεί να χρησιμοποιηθεί και να γίνει dequeue.

**Ο συγχρονισμός και η υλοποίηση του thread pool είναι ως εξής:**

Υπάρχει ένας πίνακας με τα IDs των threads που δημιουργούνται αρχικά. Επίσης υπάρχει μια ουρά στην οποία φυλάγονται τα διαθέσιμα FDs των socket που δημιουργούνται από την accept.

Όταν δημιουργείται ένα νήμα τότε κάνει lock , το κλειδώνει στο κρίσιμο τμήμα έτσι ώστε να μην εισέλθει και άλλο νήμα και cond\_wait μέχρι το queue να έχει κάτι μέσα.

Με την χρήση του thread-pool δημιουργούμε εξ αρχής ένα σταθερό αριθμό νημάτων **threadnum = 40** και όταν υπάρχει αίτηση για εξυπηρέτηση να την αναθέτει σε κάποιο από τα νήματα αυτά τα οποία δεν έχουν δουλεία. Και στην συνέχεια τα νήματα αφού εξυπηρετήσουν ένα χρήστη δεν τερματίζουν αλλά αντιθέτως μπαίνουν σε αναμονή για τις επόμενες αιτήσεις από τον χρήστη. Όπως προαναφέραμε αν δεν υπάρχει διαθέσιμο νήμα η αίτηση του πελάτη να μην είναι δυνατό να εξυπηρετηθούν, περιμένοντάς να υπάρξει κάποιο διαθέσιμο .

Για το συγχρονισμό χρησιμοποιούνται δύο σηματοφόροι (thread count & queue ) και ένα condition variable (queue).

**Η εξυπηρέτηση πελατών στο σύστημα γίνεται ως εξής:**

Αρχικά ο εξυπηρετητής υποστηρίζει μέχρι ένα αριθμό (**threadnum = 40** )από πελάτες. Όταν ο χρήστης θέλει κάποιο αρχείο να ανοίξει στο web browser διαφόρων ιστών τότε τοποθετεί 127.0.0.1:**(port)/file** και στην συνέχεια μέσα από την υλοποίηση που κάναμε του εξυπηρετητή μας εμφανίζει την ανάλογη σελίδα στον ιστό . Έτσι ο χρήστης μπορεί να καλέσει πολλές αιτήσεις και συγχρόνως να εξυπηρετηθούν.

**Δυσκολίες που αντιμετωπίστηκαν κατά τη διάρκεια της υλοποίησης περιλαμβάνουν:**

* Ένα σημαντικό πρόβλημα που αντιμετωπίσαμε είναι ο ταυτοχρονισμός των νημάτων καθώς στην αρχή δεν γνωρίζαμε πιο θα είναι το κρίσιμο τμήμα του δικού μας προγράμματος, έτσι αποφασίσαμε ότι το κρίσιμο τμήμα είναι η εισαγωγή και η εξαγωγή των νημάτων αλλά επίσης και η επεξεργασία που προκύπτει απο την ουρά (queue). Επίσης δημιουργήσαμε και μια συνάρτηση threadserve mutex και cond για την σωστή εξυπηρέτηση των αιτημάτων απο τούς χρήστες. Και τέλος χρησιμοποιήσαμε τους κατάλληλους σημαφόρους αρχικοποιώντας τους με των αριθμών των νημάτων . Αυτό το κάναμε ώστε να μπορούμε να εξυπηρετήσουμε μέχρί τα threadnum αιτήματα.
* Ένα άλλο σημαντικό μας πρόβλημα που αντιμετωπίσαμε επιτυχώς είναι η επεξεργασία των αιτημάτων απο διάφορους ιστούς σελίδων δηλαδή το περιέχόμενο του κάθε Buffer,αιτήματος ήταν διαφορετικό απο το Mozilla,firefox,chrome με αποτέλεσμα να χάναμε τις απαραίτητες πληροφορίες που θα θέλαμε όπως connection, type,filename,έτσι ώστε να στέλαμε λάθος στοιχεία να εξυπηρετούνταν τα αιτήματα μας. Στην αρχή σπάζαμε μια μια της γραμμές μας και τα τοποθετούσαμε το περιεχόμενο στην αναλογή μεταβλητή όμως όπως προαναφερθήκαμε αυτό είχε ως αποτέλεσμα να αποθηκεύονται λανθασμένες πληροφορίες και να μην μπορεί να συνεχιστεί σωστα αυτό συνέβαινε επειδή σε διαφορετικούς browsers τα στοιχεία η σειρά των στοιχείων που θέλαμε εμείς δεν ήταν με την σειρά που τα θέλαμε. Έτσι αποφασίσαμε να τα βρίσκουμε με τα όνοματα όπως δίνονται δηλαδή στο connection ,host,type,filename κτλ να τα αποθηκεύουμε στις μεταβλητές τις κατάλληλες και να μπορούμε να τα περνούμε σωστά.
* Αρχικά η υλοποίηση του thread pool βρισκόταν σε διαφορετικό αρχείο απο τον κώδικα του εξυπηρετητή που εισήγαγε στοιχεία στην ουρά και αυτό δημιουργούσε πρόβλημα στους σηματοφόρους και στα cond\_wait & cond\_signal. Αυτό οδήγησε στη μεταφορά του κώδικα του thread pool στο ίδιο αρχείο με το main κομμάτι του server για επίλυση του προβλήματος.
* Ένα ακόμα πρόβλημα που παρουσιαστηκε κατά την υλοποίηση μας είναι όταν ένας client θέλει να τρέξει ένα αίτημα όταν δεν έχει τελείωσε το προηγούμενο τότε το πρόγραμμα τερματίζει. Αν το αίτημα γίνει σε διαφορετικό tab δεν παρουσιάζεται κανένα πρόβλημα.