

Πηλιανίδης Αριστείδης

Ενσωματωμένα Συστήματα Πραγματικού χρόνου

AEM:8755

Άσκηση 3

Εισαγωγικά:

Αρχικά να πούμε ότι αναπτυχθήκαν τα προγράμματα:

- 1)client για την αποστολή μηνυμάτων(clienttask3.c)
- 2)client για την λήψη μηνυμάτων(clienttask3rec.c)
- 3)server(server3.c)

1)Client για την αποστολή μηνυμάτων

Πρώτα από όλα, το πρόγραμμα το τρέχουμε ως εξής:

./client ip port , όπου ip η διεύθυνση του ενσωματωμένου που βρίσκεται ο server και port ένα οποιοδήποτε χρησιμοποιήτο port(π.χ 2223).

Ξεκινάμε δεσμεύοντας μνήμη για τις απαραίτητες μεταβλητές, προτιμώντας το heap έναντι του stack καθώς το τελευταίο έχει περισσότερη μνήμη.

Αφού γίνει η σύνδεση με τον server ,ζητάμε από τον πελάτη να γράψει το όνομα του, σε ποιον θέλει να στείλει μήνυμα καθώς και το μήνυμα που θέλει να στείλει. Επίσης σε αυτό το σημείο μετράμε και τον

χρόνο που απαιτείται(σε ms) για να στείλει μηνύματα ο πελάτης προς τον server.

Τέλος ρωτάμε τον πελάτη αν θέλει να στείλει και άλλο μήνυμα, παίρνουμε την απάντηση του, την στέλνουμε στον server και αν είναι ναι ,επαναλαμβάνουμε την διαδικασία.

2)Client για την λήψη μηνυμάτων

Πρώτα από όλα, το πρόγραμμα το τρέχουμε ως εξής:

`./client2 ip port` , όπου ip η διεύθυνση του ενσωματωμένου που βρίσκεται ο server και port ένα οποιοδήποτε χρησιμοποιήτο port(π.χ 2223).

Ξεκινάμε δεσμεύοντας μνήμη για τις απαραίτητες μεταβλητές, προτιμώντας το heap έναντι του stack καθώς το τελευταίο έχει περισσότερη μνήμη.

Αφού γίνει η σύνδεση με τον server ,ζητάμε από τον πελάτη να γράψει το όνομα του. Έπειτα, στέλνουμε το όνομα του καθώς και την συμβολοσειρά “receiver” στον server ώστε να γνωρίζει ο server ότι ο πελάτης θέλει να λάβει μηνύματα και όχι να στείλει. Σε αυτό το σημείο μετράμε και τον χρόνο που απαιτείται(σε ms) για να λάβει μηνύματα ο πελάτης από τον server.

Τέλος, ο χρήστης λαμβάνει τα μηνύματα καθώς και τα ονόματα από όλους τους χρήστες που του έστειλαν μήνυμα ή αλλιώς ενημερώνεται ότι δεν του έστειλε κάποιος μήνυμα.

3) Server

Αρχικά να πούμε πως η υλοποίηση του server πραγματοποιήθηκε με συνδεδεμένες λίστες με κύριο στόχο τα μηνύματα που λαμβάνονται μια φορά από κάποιον χρήστη ,να διαγράφονται από την λίστα, και έτσι να εξοικονομούμε μνήμη και να διασφαλίζουμε ότι ο χρήστης δεν θα λάβει ξανά τα ήδη διαβασμένα μηνύματα του.

Για την διαγραφή και την δημιουργία νέων κόμβων όμως, χρησιμοποιήθηκε κώδικας από την ιστοσελίδα

<https://www.geeksforgeeks.org/delete-occurrences-given-key-linked-list/> . Παρατήρησα λοιπόν ότι παρόλο

που οι κόμβοι διαγράφονταν επιτυχώς, υπήρχε μια αύξηση στην virtual memory του server. Διόρθωσα πιθανά memory leaks με την χρήση του valgrind αλλά παρόλα αυτά υπήρχε ακόμα η αύξηση της μνήμης. Αυτό είχε σαν αποτέλεσμα να μην μπορούμε να τρέξουμε πάνω από 5000 threads στο ενσωματωμένο και αντίστοιχα 13000 threads στον qemu.

Όσον αφορά την δίκαιη αντιμετώπιση των χρηστών, αυτά που κάναμε για την δημιουργία δίκαιης «ουράς» είναι τα εξής:

Η συνάρτηση main δημιουργεί ένα thread που περιμένει την σύνδεση κάποιου χρήστη. Στην συνέχεια, μέσω condition variables ελέγχουμε πότε συνδέθηκε κάποιος χρήστης για να δώσουμε σήμα να δημιουργηθεί το επόμενο thread που θα εξυπηρετήσει

τον επόμενο. Βέβαια αυτό δεν σημαίνει ότι ο δεύτερος χρήστης περιμένει να εξυπηρετηθεί πλήρως ο πρώτος, παρά μόνο να συνδεθεί, έτσι ώστε να μπορεί να γίνει η εξυπηρέτηση τους παράλληλα .

Όσον αφορά την εξοικονόμηση ενέργειας έγιναν τα εξής:

- 1) Όπως αναφέραμε παραπάνω ,δημιουργούμε επιπλέον thread ,ανάλογα τους συνδεδεμένους χρήστες και έτσι δεν δημιουργούμε ταυτόχρονα πολλά αχρησιμοποίητα thread τα οποία υπερφορτώνουν το σύστημα μας και είναι πολύ πιθανό να ρίξουν τον server.
- 2) Η main βρίσκεται σε κατάσταση “sleep” μέχρι να γίνει ή σύνδεση κάποιου client.
- 3) Η εισαγωγή στην αρχή της «ουράς» έχει πολυπλοκότητα $O(1)$.

Συνεχίζοντας στο σκεπτικό του αλγορίθμου:

Εφόσον γίνει η σύνδεση κάποιου client, ενημερώνουμε μια global μεταβλητή και δίνουμε σήμα στην main να συνεχίσει με την δημιουργία και άλλου thread.

Έπειτα, διαβάζουμε το όνομα του χρήστη που στέλνει ή που θέλει να λάβει μηνύματα . Στην συνέχεια, διαβάζουμε το όνομα του χρήστη στον οποίον θέλουμε να στείλουμε μήνυμα ή την συμβολοσειρά “receiver”

και καταλαβαίνουμε ότι είμαστε σε send ή receive mode αντίστοιχα.

Αν είμαστε σε receive mode, βλέπουμε πρώτα από όλα τον χρόνο που κάνει ο server να στείλει μήνυμα στον client. Στην συνέχεια μέσω της deleteKey κάνουμε τα εξής:

- 1) Δημιουργούμε μια flag2 μεταβλητή που μας λέει αν ο χρήστης είχε να λάβει μηνύματα ή όχι, το οποίο το καταλαβαίνουμε από αναζήτηση του ονόματος του στην στήλη των Receivers όλης της ουράς . Αν δεν είχε να λάβει μηνύματα στέλνουμε το αντίστοιχο μήνυμά στον χρήστη και flag2=0. Αν είχε να λάβει μηνύματα, τότε τα στέλνουμε, και κάνουμε update την μεταβλητή flag2=1 έτσι ώστε όταν τελειώσουν τα μηνύματα του, να μπορούμε να διακόψουμε την διαδικασία.
- 2) Τα μηνύματα που στάλθηκαν τα διαγράφουμε από την ουρά και κάνουμε τα απαραίτητα update στους κόμβους της ουράς.

Αν είμαστε σε send mode, βλέπουμε πρώτα από όλα τον χρόνο που κάνει ο client να στείλει μήνυμα στον server. Στην συνέχεια , διαβάζουμε το μήνυμά που θέλει να στείλει ο client, και μέσω της push αποθηκεύουμε όλες τις πληροφορίες του client στην ουρά.

Έπειτα, διαβάζουμε την επιλογή του client να στείλει ή όχι επιπλέον μηνύματα και σε περίπτωση που έχει επιλέξει ότι θέλει να στείλει και άλλα μηνύματα επαναλαμβάνεται η διαδικασία.

Τέλος, αποδεσμεύουμε την μνήμη από τις μεταβλητές μας και κλείνουμε το file descriptor που αφορά τον πελάτη που εξυπηρετήθηκε.

Έλεγχος ορθότητας:

Για τον έλεγχο ορθότητας, ξεκινήσαμε 3 clients οι οποίοι:

- 1)Ο Α στέλνει 2 μηνύματα στον Γ
- 2)Ο Γ λαμβάνει τα μηνύματα του.(2 μηνύματα από Α)
- 3)Ο Β στέλνει 1 μήνυμα στον Γ.

Στο τέλος περιμένουμε να δούμε ότι στον server υπάρχει αποθηκευμένο μόνο το μήνυμα από τον Β προς τον Γ .

Παρακάτω βλέπουμε τις εικόνες που επιβεβαιώνουν αυτήν την συμπεριφορά(Για να είναι πιο ευδιάκριτες οι εικόνες κάντε zoom):

1)

```
fish /home/aris/Desktop/openwrt-zsun-zsun/staging_dir/toolchain-mips_mips32_gcc-4.8-
aris@aris-HP-ProBook-6450b ~/D/o/s/t/bin>
gcc -g clienttask3.c -pthread -o client
aris@aris-HP-ProBook-6450b ~/D/o/s/t/bin> ./client 192.168.1.1 2223
Please enter your name: A
Please enter the name of the client you want to send the message to: G
Please enter the message: hi
Do you want so send another message?(yes,no)
yes
Please enter your name: A
Please enter the name of the client you want to send the message to: G
Please enter the message: i am A
Do you want so send another message?(yes,no)
no
aris@aris-HP-ProBook-6450b ~/D/o/s/t/bin> █
```

```
192.168.1.1 - PuTTY
root@OpenMrt:~/aris# ./server 2223
Here is the message: A

Here is the message: G
Upload speed: 1976.00 ns
Here is the message: hi

Client wants to send another message? : yes
Here is the message: A

Here is the message: G
Upload speed: 1976.00 ns
Here is the message: i am A

Client wants to send another message? : no
█
```

2)

```
fish /home/aris/Desktop/openwrt-zsun-zsun/staging_dir/toolchain-mips_mips32_gcc-4.8-
Please enter your name: A
Please enter the name of the client you want to send the message to: G
Please enter the message: hi
Do you want so send another message?(yes,no)
yes
Please enter your name: A
Please enter the name of the client you want to send the message to: G
Please enter the message: i am A
Do you want so send another message?(yes,no)
no
aris@aris-HP-ProBook-6450b ~/D/o/s/t/bin> ./client2 192.168.1.1 2223
Please enter your name: G

Checking if you got mail...
Download speed: 1976.00 ns

Message: hi
from: A

Message: i am A
from: A

aris@aris-HP-ProBook-6450b ~/D/o/s/t/bin> █
```

```
192.168.1.1 - PuTTY
Here is the message: A

Here is the message: G
Upload speed: 1976.00 ns
Here is the message: hi

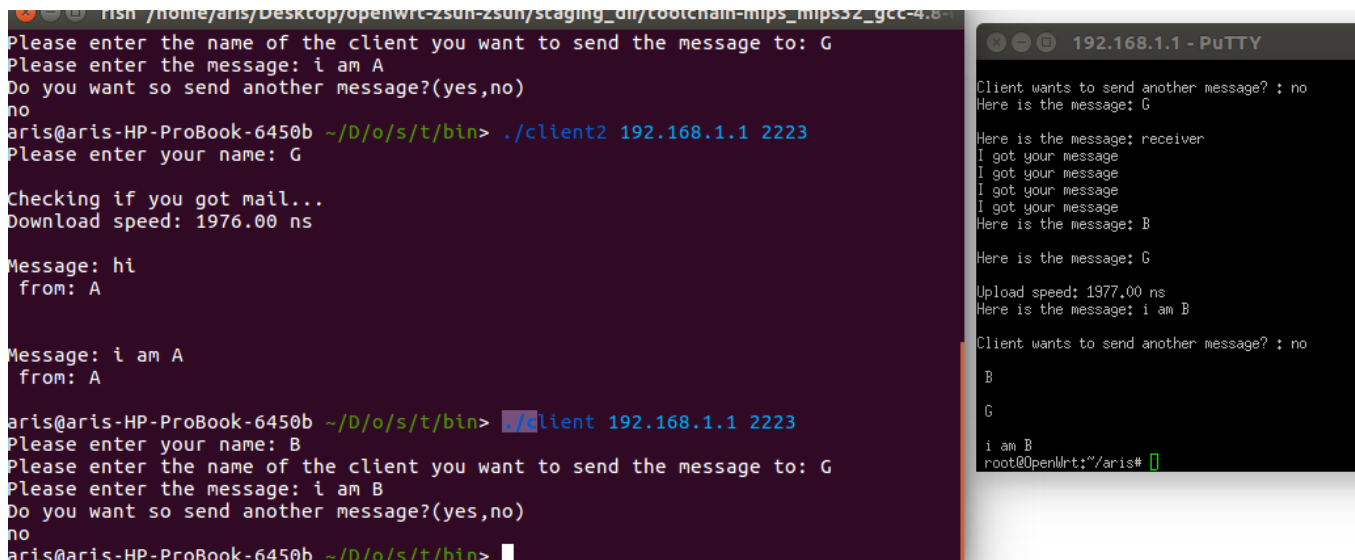
Client wants to send another message? : yes
Here is the message: A

Here is the message: G
Upload speed: 1976.00 ns
Here is the message: i am A

Client wants to send another message? : no
Here is the message: G

Here is the message: receiver
I got your message
I got your message
I got your message
I got your message
█
```

3)



```
fish /home/aris/Desktop/openwrt-zsdu-zsdu/staging_dir/toolchain-mips_mips32_gcc-4.8...
Please enter the name of the client you want to send the message to: G
Please enter the message: i am A
Do you want so send another message?(yes,no)
no
aris@aris-HP-ProBook-6450b ~/D/o/s/t/bin> ./client2 192.168.1.1 2223
Please enter your name: G

Checking if you got mail...
Download speed: 1976.00 ns

Message: hi
from: A

Message: i am A
from: A

aris@aris-HP-ProBook-6450b ~/D/o/s/t/bin> ./client 192.168.1.1 2223
Please enter your name: B
Please enter the name of the client you want to send the message to: G
Please enter the message: i am B
Do you want so send another message?(yes,no)
no
aris@aris-HP-ProBook-6450b ~/D/o/s/t/bin>

192.168.1.1 - PuTTY
Client wants to send another message? : no
Here is the message: G

Here is the message: receiver
I got your message
I got your message
I got your message
I got your message
Here is the message: B

Here is the message: G

Upload speed: 1977.00 ns
Here is the message: i am B

Client wants to send another message? : no
B
G
i am B
root@OpenMrt:~/aris#
```

Χρόνος αποστολής/λήψης μηνυμάτων:

Όπως βλέπουμε από παραπάνω ο χρόνος αποστολής/λήψης για χαμηλό φόρτο είναι περίπου 2000 ns δηλαδή 2 μ s. Η μέτρηση έγινε με την `clock_gettime` με παράμετρο `CLOCK_MONOTONIC`, ώστε να μην επηρεάζεται από πιθανές αλλαγές στο ρολόι του συστήματος μας.

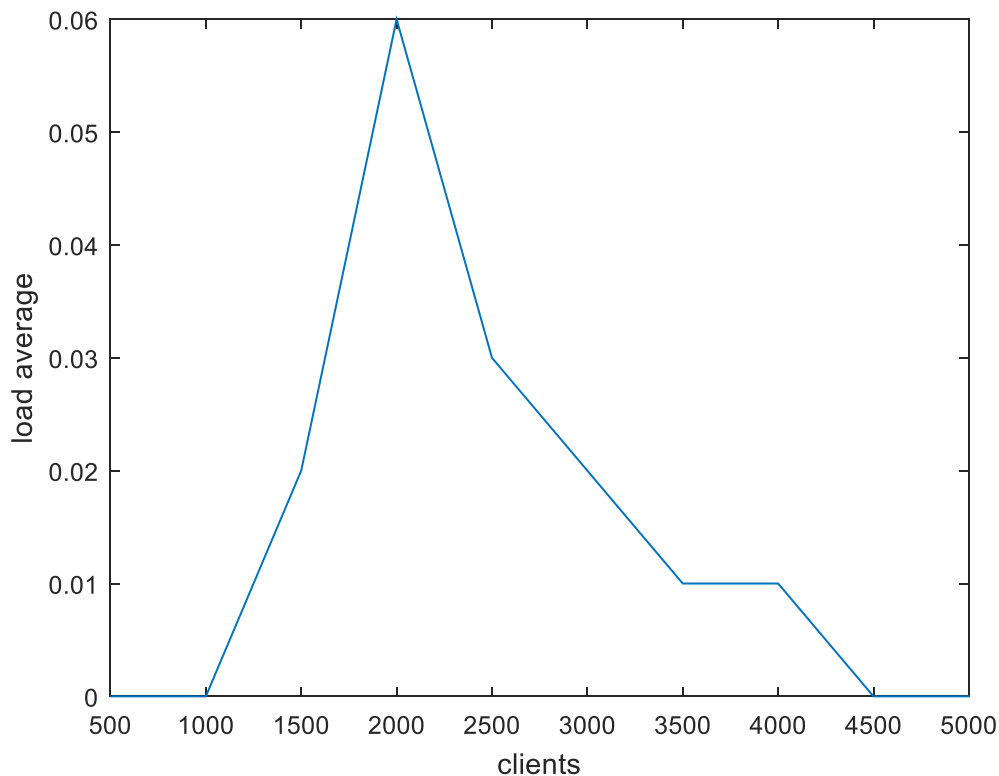
Κατανάλωση ενέργειας:

Για την κατανάλωση ενέργειας βάλαμε τον server να τρέχει στο background και με την εντολή `top` παρακολουθούσαμε το Load average. Ακόμη με την εντολή `top` βλέπουμε και την Virtual Memory που χρησιμοποιεί ο server.

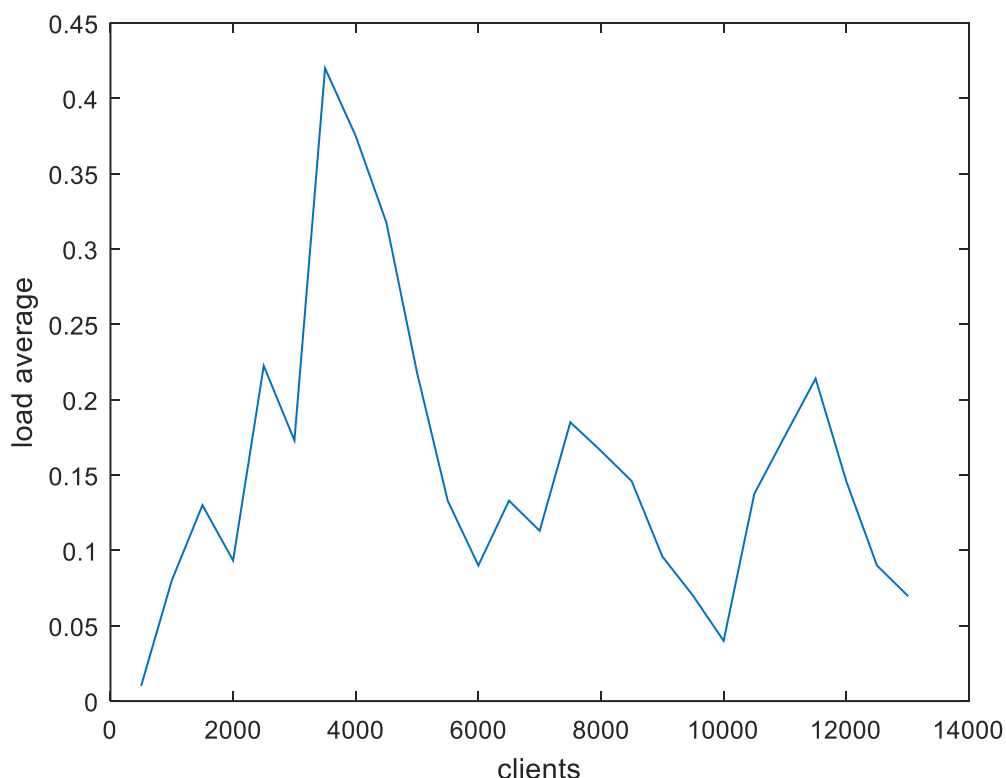
Προσοχή!!! Είναι σημαντικό πριν τρέξει ο server, να τρέξετε την εντολή `ulimit -s 8` η οποία αλλάζει το όριο του stack size που μπορεί να κάνει allocate κάθε thread(default 8192) διαφορετικά είναι πολύ πιθανό να ξεμείνει ο server από virtual memory.

Έγινε έλεγχος της κατανάλωσης για πλήθος χρηστών από 500 έως 5000 clients στο zsun για τον λόγο που εξηγήθηκε παραπάνω και έγιναν οι μετρήσεις και στον qemu (μέχρι 13000 threads) που μπορούσαμε να έχουμε μεγαλύτερο εύρος από thread. Πήραμε την μέση τιμή load average για κάθε πείραμα και τα αποτελέσματα φαίνονται στα παρακάτω γραφήματα :

Zsun:



Qemu:



Αρά ο βέλτιστος λόγος (αριθμός μηνυμάτων/W) στο zsh φαίνεται να είναι στους 5000 πελάτες ενώ στον qemu που έχουμε μεγαλύτερο εύρος βλέπουμε ότι ο βέλτιστος λόγος είναι στους 10000 πελάτες.

Dropbox link:

<https://www.dropbox.com/sh/k6bexx0cy9pyt4i/AABHcYP8u53sZ66LpCD7Uc0-a?dl=0>