Γεωργούλας Βασίλης Α.Μ.: 2954
Βάμβας Ιωάννης Α.Μ.: 2943

Υλοποιηση 1ης εργαστηριακης ασκησης

Πολυνηματικη υλοποιηση

Αρχειο server.c

Στην συναρτηση main βρισκεται η υλοποιηση του παραγωγου(ουσιαστικα ο κωδικας της while που βρισκεται στην μαιν). Πριν την while παραγονται τα νηματα του καταναλωτη(που θελουμε να δημιουργηθουν εκ των προτερων).

Στην συναρτηση *thread function βρισκεται η υλοποιηση του κωδικα των καταναλωτων οι οποιοι εξυπηρετουν τις αιτησεις που φτανουν στον server. Επισης και εδω ο κωδικας των καταναλωτων ειναι μεσα σε while.

Διαχειρηση ουρας και ελεγχος ταυτοχρονισμου

Αρχειο server.c

Παραγωγος-Καταναλωτες

Στον κωδικα του παραγωγου αρχικα δεχομαστε τις καινουργιες αιτησεις, μετα κανουμε lock(&mutexP) τον παραγωγο ωστε να εξασφαλισουμε ότι μονο 1 παραγωγος θα ειναι μεσα στην ουρα, μετεπειτα ελεγχουμε αν η ουρα ειναι γεματη(δηλαδη size==QUEUE_SIZE) και αν οντως η ουρα ειναι γεματη τοτε περιμενουμε(με cond wait &not_ full) μεχρι να βρεθει κενη θεση. Στην συνεχεια περναμε στην ουρα ενα struct που περιεχει τον περιγραφεα αρχειου και την χρονικη στιγμη(σε μικροδευτερολεπτα) που μπηκε η αιτηση στην ουρα. Επιπλεον εχουμε 1 μετρητη size ο οποιος υποδηλωνει ποσα στοιχεια εχουμε στην ουρα. Τελος στελνουμε(με signal &not_empty) στους καταναλωτες οτι μπηκε μια αιτηση στην ουρα και κανουμε unlock(&mutexP) παραγωγο.

Στον κωδικα του καταναλωτη(δηλαδη στην συναρτηση thread_function) τρεχουμε μια while και μετα κανουμε lock(&mutexC). Στη συνεχεια οσο η ουρα ειναι αδεια(δηλαδη size==0) τοτε οι καταναλωτες περιμενουν να ερθει μια καινουργια αιτηση με cond_wait(&not_empty). Μολις φτασει η καινουργια αιτηση, ενας απο τους καταναλωτες θα παει και θα βγαλει απο την ουρα τον περιγραφεα αρχειου αλλα και την χρονικη στιγμη που η αιτηση μπηκε στην ουρα, επιπροσθετα μειωνουμε το size κατα 1 αφου βγαζουμε 1 στοιχειο απο την ουρα και στελνουμε σημα(με signal &not_ full) οτι υπαρχει 1 διαθεσιμη θεση στην ουρα για την υποδοχη μιας καινουργιας αιτησης και μετα θα γινει unlock(& mutexC). Τελος θα γινει η εξυπηρετηση της αιτησης με την εντολη process_request(εξω απο το unlock ωστε να εξυπηρετουνται πολλαπλες αιτησεις) και κλεινουμε τον περιγραφεα με την εντολη close.

Υλοποιηση κυκλικης ουρας

Αρχικα, η υλοποιηση της ουρας εγινε με 1 πινακα στον οποιο καθε φορα βαζουμε 1 struct οπου περιεχεται η χρονικη στιγμη που εφτασε η αιτηση και ο περιγραφες αρχειου. Αρχικοποιουμε τα head και tail στο -1. Ο παραγωγος βαζει στοιχεια στην ουρα και αυξανει το tail κυκλικα και ο καταναλωτης βγαζει στοιχεια απο την ουρα και αυξανει το head κυκλικα. Οταν βαζουμε το 1ο στοιχειο στην ουρα τοτε το head γινεται ισο με 0 ενω οταν βγαζουμε το τελευταιο στοιχειο της ουρας τοτε το head και το tail γινονται -1.

Αναγνωστες-Γραφεις (προτεραιοτητα στους αναγνωστες)

Αναγνωστες(GET)

Αρχικα κανουμε lock(με &readwrite_mutex) και μετα οσο υπαρχουν γραφεις μεσα στην βαση(writer_count>0) τοτε περιμενουμε με wait(&reader_cond) μεχρι να μην υπαρχει κανενας γραφεας μεσα στην βαση, αν δεν υπαρχει κανενας γραφεας μεσα στην βαση τοτε αυξανουμε τον αριθμο των αναγνωστων(reader_count). Μετα κανουμε unlock(& readwrite_mutex) ωστε να μπορουν να γινουν παραλληλα GET, μολις ολοκληρωθουν τα GET τοτε ξανακανουμε lock(&readwrite_mutex) και μειωνουμε τον αριθμο των αναγνωστων(reader_count) και ελεγχουμε αν δεν υπαρχει καποιος αναγνωστης μεσα στην βαση(αν οντως δεν υπαρχει αναγνωστης μεσα στην βαση τοτε στελνουμε σημα σε καποιο γραφεα που περιμενει με signal(&writer_cond) και μετα κανουμε unlock(&readwrite_mutex).

Γραφεις(PUT)

Κανουμε lock(&readwrite_mutex) και ελεγχουμε αν υπαρχουν γραφεις ή αναγνωστες μεσα στην βαση(writer_count>0||reader_count>0) και περιμενουμε με wait(&writer_cond) μεχρι να μην ειναι κανενας αναγνωστης και κανενας γραφεας μεσα στην βαση. Στη συνεχεια το writer_count γινεται 1 δηλαδη υπαρχει 1 γραφεας μεσα στην βαση, εκτελειτε το PUT, μετα το writer_count γινεται 0 και στελνουμε σημα στον επομενο γραφεα που περιμενει(signal &writer_cond), στελνουμε σημα με broadcast σε ολους τους αναγνωστες που περιμενουν(broadcast &reader_cond). Τελος κανουμε unlock(&readwrite_mutex).

Σηματα και χρονομετρηση

Αρχειο server.c

Το **σημα** ctrl+Z το χειριζομαστε με την συναρτηση sighandler και ο ορισμος γινεται μεσα στην main() με την εντολη signal(SIGTSTP, sighandler). Μετα δινουμε τις σχετικες παραμετρους για την σωστη αναθεση του σηματος στην συναρτηση sighandler. Μεσα στην συναρτηση sighandler κανουμε lock(&mutex) μετα αλλαζουμε το flag και το κανουμε false ωστε να τερματιστουν οι while και των καταναλωτων και του παραγωγου. Επισης εχουμε μια συνθηκη !flag μεσα στο new_fd==-1 ωστε να πιασουμε το σημα σωστα και να τερματισει ομαλα ο παραγωγος(σε οποιοδηποτε αλλο λαθος θα

τερματισει με error(accept()) οπως πρεπει. Στη συνεχεια μεσα στην συναρτηση sighandler κανουμε broadcast(&not empty) ωστε να ξυπνησουμε ολους τους καταναλωτες και μετα κανουμε unlock(&mutex). Τελος κανουμε join ωστε να τερματισουν ομαλα ολα τα νηματα καταναλωτη. Τελος το flag υπαρχει και μεσα στον καταναλωτη ώστε να εξασφαλιζουμε τον σωστο τερματισμο του καταναλωτη. Επιπροσθετα υπαρχει χρηση της συναρτησης pthread_sigmask ώστε να μπλοκαρουμε το σημα ctrl+Z για τους καταναλωτες(και χρησιμοποιειτε ακριβως πριν την δημιουργια των καταναλωτων δηλαδη πριν τα pthread_create) και μετα το τελος της δημιουργιας των νηματων καταναλωτη ξεμπλοκαρουμε το σημα ctrl+Z ώστε να θεσουμε την συναρτηση που θα χειριστει αυτό το σημα στον παραγωγο.

Για την **χρονομετρηση** εχουμε τις 2 κοινοχρηστες μεταβλητες total_waiting_time και total_service_time. Για το total_waiting_time μεσα στον καταναλωτη περνουμε το στοιχειο της ουρας που περιεχει την χρονικη στιγμη που η αιτηση μπηκε στην ουρα και μολις η αιτηση βγει απο την ουρα περνουμε την συγκεκριμενη χρονικη στιγμη με την εντολη gettimeofday(&wait2, NULL).

Για το total_service_time εντελως αναλογα με απο πανω, περνουμε την χρονικη στιγμη πριν την εκτελεση της εντολης process_request και την χρονικη στιγμη ακριβως μετα την εκτελεση της εντολης process_request.

Οι χρονοι που υπολογιζουμε ειναι σε μικροδευτερολεπτα και μεσα στην main οταν τυπωνουμε τα στατιστικα του προγραμματος τους μετατρεπουμε σε δευτερολεπτα.

Επιπλεον εχουμε τις κοινοχρηστες μεταβλητες completed_requests(η οποια αυξανεται μετα το τελος της process_request) και ειναι οι συνολικες αιτησεις που εχουν εξυπηρετηθει και βαλαμε 2 εξτρα μεταβλητες(τον μεσο χρονο αναμονης και τον μεσο χρονο εξυπηρετησης) που στηριχθηκαν στους τυπους total_waiting_time/completed_requests και total_service_time/completed_requests.

Τελος επειδη οι μεταβλητες total_waiting_time, total_service_time, completed_requests ειναι κοινοχρηστες προστατευονται με mutex κατα την ενημερωση τους.

Αρχειο client.c

Δημιουργονται πολλαπλα νηματα που καθε ενα απο αυτα καλει την talk επαναληπτικα(MAX_STATION_ID φορες) και αναλογα με το i στην φορ που γινονται τα πολλαπλα νηματα(αν το i ειναι αρτιος τοτε το count είναι 0 και περναμε το count στην thread function ωστε να εκτελεσει get ενω αν το i ειναι περιττος τοτε το count είναι 1 και περναμε το count στην thread function ωστε να γινει put). Μεσα στην thread_function εγκαθιδρυουμε συνδεση σοκετ ωστε να γινει το talk σωστα και τα put και τα get που εκτελουνται παραλληλα εκτελεουνται .Αναλητικοτερα γινονται put-get εναλλαξ ανα νημα, αν εχουμε count = 0 τοτε ξεκιναμε με get συνεχιζουμε με put και παει λεγοντας, ενώ αν εχουμε count=1 τοτε ξεκιναμε με put συνεχιζουμε με get και παει λεγοντας.Τελος τα put/get συνεχως εκτελουνται πανω στον ιδιο σταθμο.

Μετρησεις

Οι μετβλητες των μετρησεων ειναι ο αριθμος των καταναλωτων(NTHREADS) και το μεγεθος της ουρας(QUEUE_SIZE), τα νηματα που κανουν τις αιτησεις και ποσες αιτησεις στελνει κάθε νημα. Στα γραφηματα η κοκκινη γραμμη αντιστιχει στο total waiting time ενω η μαυρη στο the total service time.

---

20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...5 νηματα καταναλωτη και μεγεθος ουρας 5

the completed_requests are 220
the total waiting time was: 0.036491
the total service time was 0.010408

20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...10 νηματα καταναλωτη και μεγεθος ουρας 5

the completed_requests are 220
the total waiting time was: 0.124098
the total service time was 0.016533

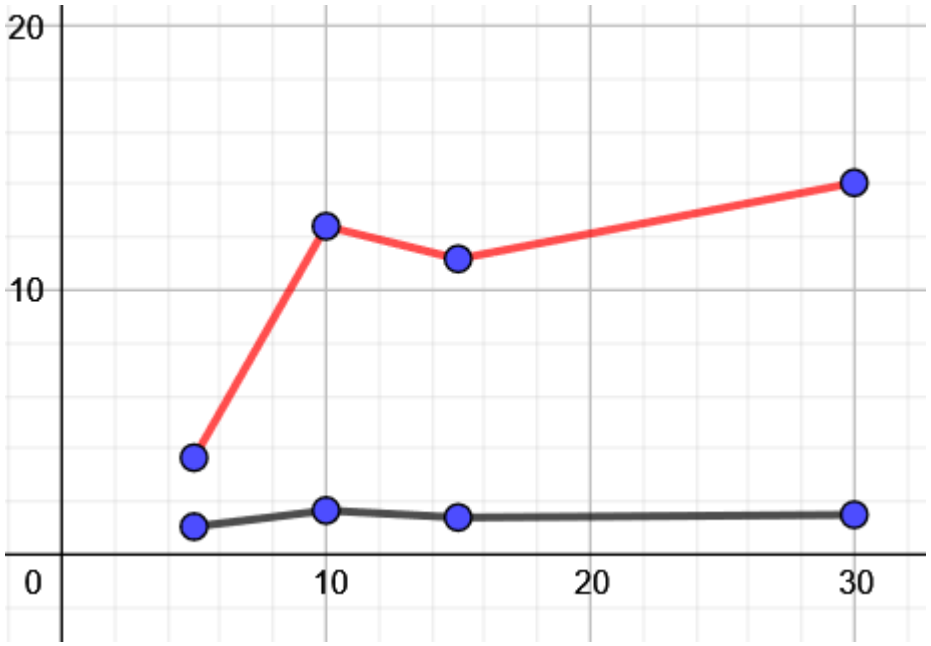
20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...15 νηματα καταναλωτη και μεγεθος ουρας 5

the completed_requests are 220
the total waiting time was: 0.111746
the total service time was 0.013877

20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...30 νηματα καταναλωτη και μεγεθος ουρας 5

the completed_requests are 220
the total waiting time was: 0.140557

the total service time was 0.014899



(x10)

αξονας y waiting time/service time,αξονας x νηματα καταναλωτη

---

20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...5 νηματα καταναλωτη και μεγεθος ουρας 5

the completed_requests are 220
the total waiting time was: 0.037354
the total service time was 0.011992

20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...5 νηματα καταναλωτη και μεγεθος ουρας 10

the completed_requests are 220
the total waiting time was: 0.063478
the total service time was 0.010754

20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...5 νηματα καταναλωτη και μεγεθος ουρας 15

the completed_requests are 220
the total waiting time was: 0.045531
the total service time was 0.011799

20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...5 νηματα καταναλωτη και μεγεθος ουρας 20

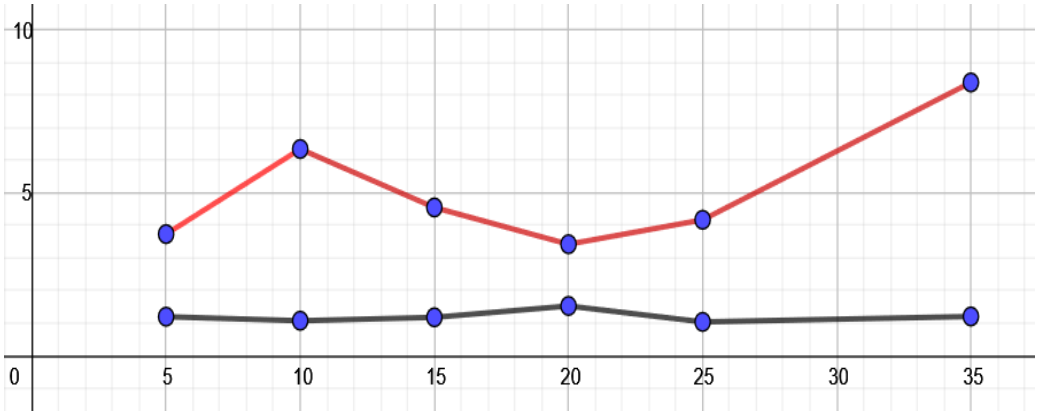
the completed_requests are 220
the total waiting time was: 0.034250
the total service time was 0.015290

20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...5 νηματα καταναλωτη και μεγεθος ουρας 25

the completed_requests are 220
the total waiting time was: 0.041739
the total service time was 0.010397

20 νηματα, καθε 1 απο αυτα στελνει 11 αιτησεις(put-get παραλληλα)...5 νηματα καταναλωτη και μεγεθος ουρας 35

the completed_requests are 220
the total waiting time was: 0.083916
the total service time was 0.012060



(x10)
αξονας y waiting time/service time,αξονας x μεγεθος ουρας

<u>ΣΧΟΛΙΟ</u> :

- Το μικροτερο total waiting time επιτυγχανεται στο τεστ με 5 νηματα καταναλωτη και μεγεθος ουρας 20(total requests 220).
- Το μικροτερο total service time επιτυγχανεται στο τεστ με 5 νηματα καταναλωτη και μεγεθος ουρας 25. (total requests 220).

---

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...1 νημα καταναλωτη και μεγεθος ουρας 5.

the completed_requests are 100
the total waiting time was: 0.173142
the total service time was 0.028616

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...3 νηματα καταναλωτη και μεγεθος ουρας 5

the completed_requests are 100
the total waiting time was: 0.027643
the total service time was 0.008258

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...5 νηματα καταναλωτη και μεγεθος ουρας 5

the completed_requests are 100
the total waiting time was: 0.017333
the total service time was 0.008128

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...7 νηματα καταναλωτη και μεγεθος ουρας 5

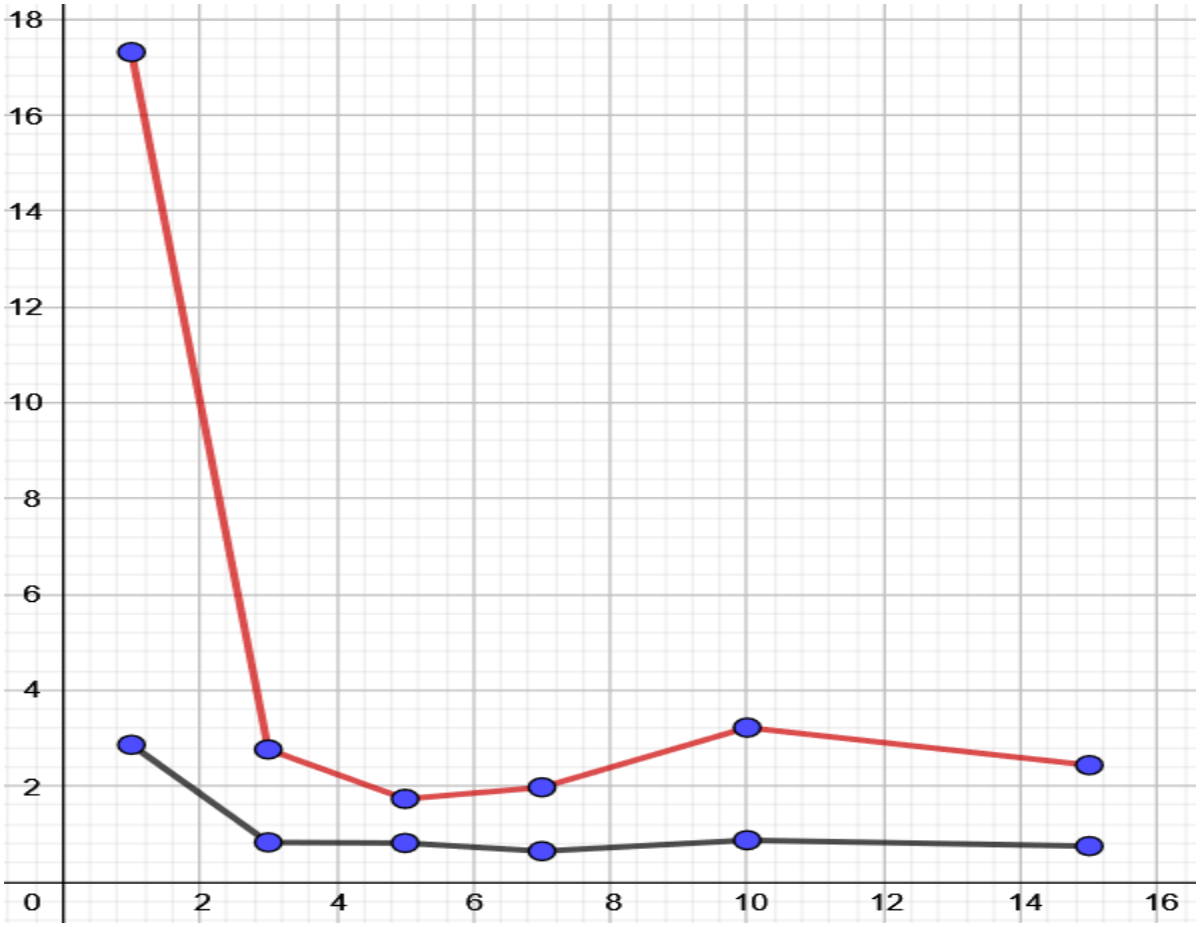
the completed_requests are 100
the total waiting time was: 0.019746
the total service time was 0.006430

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...10 νηματα καταναλωτη και μεγεθος ουρας 5

the completed_requests are 100
the total waiting time was: 0.032208
the total service time was 0.008710

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...15 νηματα καταναλωτη και μεγεθος ουρας 5

the completed_requests are 100
the total waiting time was: 0.024365
the total service time was 0.007482



(x10)
αξονας y waiting time/service time,αξονας x νηματα καταναλωτη

---

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...3 νηματα καταναλωτη και μεγεθος ουρας 2

the completed_requests are 100
the total waiting time was: 0.019599
the total service time was 0.009877

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...3 νηματα καταναλωτη και μεγεθος ουρας 5

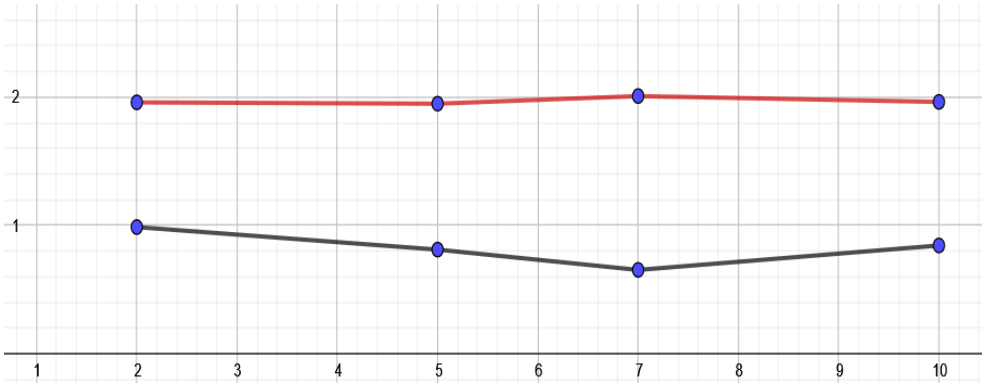
the completed_requests are 100
the total waiting time was: 0.019496
the total service time was 0.008119

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...3 νηματα καταναλωτη και μεγεθος ουρας 7

the completed_requests are 100
the total waiting time was: 0.020091
the total service time was 0.006536

10 νηματα, καθε απο 1 αυτα στελνει 10 αιτησεις(put-get παραλληλα)...3 νηματα καταναλωτη και μεγεθος ουρας 10

the completed_requests are 100
the total waiting time was: 0.019634
the total service time was 0.008442



(x10)
αξονας y waiting time/service time,αξονας x μεγεθος ουρας

ΣΧΟΛΙΟ :

- Το μικροτερο total waiting time επιτυγχανεται στο τεστ με 5 νηματα καταναλωτη και μεγεθος ουρας 5 (total requests 100).
- Το μικροτερο total service time επιτυγχανεται στο τεστ με 7 νηματα καταναλωτη και μεγεθος ουρας 5. (total requests 100).