Λειτουργικά Συστήματα
ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΉ ΆΣΚΗΣΗ 1

Σιρινιάν Αράμ Εμμανουήλ
ΑΜ: 2537
ae.sirinian@gmail.com

Μάρτιος 2019

Τι έχει υλοποιηθεί

Στον κώδικα που παραδόθηκε υλοποιήθηκε ένας απλός πολυνηματικός διακομιστής αποθήκευσης ζευγών κλειδιού-τιμής. Το λογισμικό δέχεται μια μικρή ακολουθία από ζεύγη κλειδιού τιμής για αποθήκευση στο διακομιστή και κλειδιά για ταυτόχρονη αναζήτηση. Επίσης διατηρεί στατιστικά σχετικά με τους χρόνους εξυπηρέτησης των αιτήσεων. Ο κώδικας εκμεταλλεύεται τη βιβλιοθήκη νημάτων επιπέδου χρήστη POSIX threads του συστήματος Linux.

Χρησιμοποιείται η προσέγγιση πελάτη-διακομιστή και η βασική λειτουργικότητα ήταν έτοιμη από ένα συνοδευτικό αρχείο myy601Lab1.zip.

Περιγραφή της λειτουργίας του πολυνηματικού διακομιστή αποθήκευσης

Ο διακομιστής έχει δομή παραγωγού-καταναλωτή. Ένα νήμα παραγωγού περιμένει για εισερχόμενες αιτήσεις σύνδεσης, ενώ ένα προκαθορισμένο πλήθος από νήματα καταναλωτή είναι υπεύθυνα για την εξυπηρέτηση των αιτήσεων αναζήτησης. Αρχικά υπάρχει μόνο το νήμα παραγωγού, το αρχικό νήμα που δημιουργείται.

Στην αρχή της εκτέλεσης του δημιουργείται μια ουρά σταθερού μέγιστου μεγέθους στην οποία εισάγονται οι περιγραφείς αρχείων καθώς και οι χρόνοι έναρξης τους. Στην συνέχεια γίνονται αρχικοποιήσεις πολλών κοινόχρηστων μεταβλητών. Μετά καθορίζονται οι ενέργειες που θα συμβούν με την παράδοση ενός σήματος SIGSTP ή SIGINT. Μια από αυτές τις ενέργειες είναι να ενημερώσουν τα νήματα καταναλωτή να τελειώσουν και στην περίπτωση του SIGSTP να εκτυπωθεί το πλήθος των αιτήσεων που έχουν εξυπηρετηθεί, το μέσο χρόνο αναμονής, καθώς και το μέσο χρόνο εξυπηρέτησης των αιτήσεων ώστε στο τέλος να τερματίσει. Ύστερα το νήμα παραγωγού κάνει διάφορες ενέργειες σχετικές με την δημιουργία της σύνδεσης με τον πελάτη και την βάση KISSDB. Σε αυτό το σημείο

δημιουργούνται τα νήματα καταναλωτή σύμφωνα από έναν προκαθορισμένο αριθμό τα οποία και αποδεσμεύονται.

Ακολουθεί ένας ατέρμονος βρόχος στον οποίο ο παραγωγός περιμένει για συνδέσεις και όταν αυτές έρχονται εκτελεί μια μέθοδο που προσθέτει τον περιγραφέα σύνδεσης καθώς και τον χρόνο έναρξης της στην κοινόχρηστη ουρά. Παρομοίως ένας βρόχος υπάρχει στα νήματα καταναλωτή τα οποία αφαιρούν από την ουρά έναν περιγραφέα και τον χρόνο έναρξης. Μετά υπολογίζουν και προσθέτουν τον συνολικό χρόνο αναμονής καθώς και το άθροισμα των συνολικών χρόνων αναμονής. Τέλος ενημερώνουν το πλήθος των αιτήσεων που έχουν εξυπηρετηθεί. Επειδή όμως η ουρά, οι χρόνοι και το πλήθος αιτήσεων είναι κοινόχρηστες πρέπει να διασφαλιστεί ο συγχρονισμός των νημάτων. Το οποίο και επιτυγχάνεται με κλειδαριές αμοιβαίου αποκλεισμού και μεταβλητές συνθήκες. Μια κλειδαριά για την εξασφάλιση ότι παραγωγός και καταναλωτές δεν πειράζουν την ουρά ταυτόχρονα. Μια κλειδαριά και μια μεταβλητή συνθήκη για τον έλεγχο πότε μια ουρά αδειάζει και ανάλογα όταν γεμίζει.

Η τελευταία δουλειά που καλούνται να πραγματοποιήσουν τα νήματα καταναλωτή είναι η εκτέλεση της αίτησης του πελάτη. Η δουλειά αυτή γίνεται με παράλληλο τρόπο. Πιο συγκεκριμένα ανοίγει η αποθήκη και εφαρμόζετε η αιτούμενη λειτουργία PUT ή GET με το προκαθορισμένο ζεύγος κλειδιού-τιμής ή κλειδί και αυτό με παράλληλο τρόπο. Οι αιτήσεις PUT εκτελούνται ακολουθιακά η μια μεταξύ της άλλης ενώ οι αιτήσεις GET παράλληλα η μία μεταξύ της άλλης.

Αλλαγές στον κώδικα του πελάτη

Ο κώδικας του πελάτη έχει τροποποιηθεί με τέτοιο τρόπο ώστε να μπορεί επιπρόσθετα να στέλνει PUT και GET ταυτόχρονα. Ο client έχει μια προκαθορισμένη τιμή για το πόσα fork θα κάνει (#define NUMBER_OF_FORKS). Προσοχή, δεν είναι ο αριθμός διεργασιών αλλά των forks. Το κάθε fork στέλνει επαναλαμβανόμενα αιτήσεις PUT και GET στον πελάτη, μέχρι κάποιον μέγιστο αριθμό ο οποίος επίσης είναι προκαθορισμένος.

Παράδειγμα εκτέλεσης:

./client -a localhost -i 1 -x

Εξφαλμάτωση του κώδικα

Για την εξφαλμάτωση του κώδικα ελέγχθηκε αν ο αμοιβαίως αποκλεισμός μεταξύ των κοινόχρηστων μεταβλητών των νημάτων δουλεύει σωστά. Επίσης όλες οι απαιτήσεις για τις λειτουργίες πληρούνται.

Ελέγχθηκε ότι όταν το νήμα παραγωγού εισάγει ένα περιγραφέα σύνδεσης και τον χρόνο εισαγωγής του. Υπάρχει χώρος στην ουρά και κανένα νήμα καταναλωτή δεν την πειράζει. Παρομοίως για όλες τις εμπλεκόμενες κοινόχρηστες μεταβλητές.

Όταν ένα νήμα καταναλωτή θέλει να εξάγει από την ουρά ελέγχθηκε ότι κανένα άλλο νήμα παραγωγού ή καταναλωτή δεν την πειράζουν αλλά και οι κοινόχρηστες μεταβλητές που αφορούν αυτήν την λειτουργία. Η εξαγωγή πραγματοποιείται

εφόσον η ουρά δεν είναι άδεια. Δόθηκε μεγάλη προσοχή ώστε δύο νήματα καταναλωτή που τρέχουν παράλληλα να μην μπορούν να ακολουθήσουν μια ουρά με λάθος τρόπο και να παίρνουν τιμές σκουπίδια.

Τέλος έχει ελεγχθεί πως όταν ο κώδικας διακομιστή ανοίγει την αποθήκη και εφαρμόζει την αιτούμενη λειτουργία PUT ή GET με το καθορισμένο ζεύγος κλειδιού-τιμής ή κλειδί, εξασφαλίζονται οι απαιτήσεις για τον σωστό ταυτοχρονισμό. Πιο συγκεκριμένα, ο διακομιστής υποστηρίζει ταυτόχρονη εκτέλεση των PUT και GET. Δόθηκε προσοχή ώστε οι αιτήσεις PUT να εκτελούνται ακολουθιακά αλλά και να υποστηρίζονται πολλαπλές λειτουργίες GET ταυτόχρονα τη μία σε σχέση με τις άλλες.

Ο κώδικας ελεγχόταν με εντολές εκτύπωσης αλλά και με τον gdb debugger. Στις παρακάτω εικόνες φαίνεται πως βρέθηκε ένα λάθος στον τρόπο που δύο νήματα καταναλωτή προσπερνούσαν την δομή ουράς στον κώδικα του διακομιστή.

```
dequeue new_fd: 6, head: 7, tail: 6, item_count: 0
Thread 140344473421568 PUT socket_fd: 6
(Info!!!!!) main: Got connection from '127.0.0.1' new_fd: 5
non_empty_queue: true, non_full_queue: true
enqueue new_fd: 5, head: 7, tail: 7, item_count: 1
non_empty_queue: false, non_full_queue: true
(Info!!!!!) main: Got connection from '127.0.0.1' new_fd: 6
dequeue new_fd: 5, head: 0, tail: 7, item_count: 0
non_empty_queue: true, non_full_queue: true
enqueue new_fd: 6, head: 0, tail: 0, item_count: 1
Thread 140344465028864 PUT socket_fd: 5
(Info!!!!!) main: Got connection from '127.0.0.1' new_fd: 7
non_empty_queue: true, non_full_queue: true
enqueue new_fd: 7, head: 0, tail: 1, item_count: 2
non_empty_queue: true, non_full_queue: true
dequeue new_fd: 6, head: 1, tail: 1, item_count: 1
Thread 140344490206976 PUT socket_fd: 6
non_empty_queue: false, non_full_queue: true
dequeue new_fd: 7, head: 2, tail: 1, item_count: 0
Thread 140344465028864 PUT socket_fd: 7
(Info!!!!!) main: Got connection from '127.0.0.1' new_fd: 8
non_empty_queue: true, non_full_queue: true
enqueue new_fd: 8, head: 2, tail: 2, item_count: 1
(Info!!!!!) main: Got connection from '127.0.0.1' new_fd: 5
non_empty_queue: false, non_full_queue: true
dequeue new_fd: 8, head: 3, tail: 2, item_count: 0
Thread 140344473421568 PUT socket_fd: 8
non_empty_queue: true, non_full_queue: true
enqueue new_fd: 5, head: 3, tail: 3, item_count: 1
(Info!!!!!) main: Got connection from '127.0.0.1' new_fd: 6
non_empty_queue: true, non_full_queue: true
enqueue new_fd: 6, head: 3, tail: 4, item_count: 2
non_empty_queue: true, non_full_queue: true
(Info!!!!!) main: Got connection from '127.0.0.1' new_fd: 7
dequeue new_fd: 5, head: 4, tail: 4, item_count: 1
non_empty_queue: false, non_full_queue: true
dequeue new_fd: 6, head: 5, tail: 4, item_count: 0
Thread 140344490206976 PUT socket_fd: 5
non_empty_queue: true, non_full_queue: true
dequeue new_fd: 5, head: 6, tail: 4, item_count: -1
Thread 140344481814272 PUT socket_fd: 6
non_empty_queue: true, non_full_queue: true
dequeue new_fd: 6, head: 7, tail: 4, item_count: -2
non_empty_queue: true, non_full_queue: true
dequeue new_fd: 5, head: 0, tail: 4, item_count: -3
Error in read_from_socket(). Cause: Bad file descriptor
[aram@TZif2 osNewProjectV5]$
```

Βασικές δομές και συναρτήσεις που χρησιμοποιήθηκαν

struct fdStartTimeCouple_s:	Κρατάει έναν περιγραφέα σύνδεσης καθώς και τον χρόνο έναρξης.
struct myQueue_s:	Η δομή ουρά.
void producer_thread_function(int new_fd):	Η συνάρτηση που καλεί το νήμα παραγωγού για να εισάγει στην κοινόχρηστη ουρά έναν νέο περιγραφέα σύνδεσης καθώς και τον χρόνο έναρξης.
void *consumer_thread_function():	Η συνάρτηση που καλεί το νήμα καταναλωτή για να εξάγει από την κοινόχρηστη ουρά έναν περιγραφέα σύνδεσης καθώς και τον χρόνο έναρξης. Επίσης υπολογίζει τον συνολικό χρόνο αναμονής και εξυπηρέτησης της αίτησης, τα αθροίσματα όλων τους από όλες τις συνδέσεις και το πλήθος των αιτήσεων που εξυπηρετήθηκαν.
double compute_total_waiting_time(fdStartTimeCouple_t dequeued_item):	Η συνάρτηση αυτή υπολογίζει τον συνολικό χρόνο αναμονής και συνολικό χρόνο εξυπηρέτησης των αιτήσεων.
double compute_total_service_time(struct time start_of_service_time):	Η συνάρτηση αυτή υπολογίζει τον συνολικό χρόνο αναμονής και συνολικό χρόνο εξυπηρέτησης των αιτήσεων.
void enqueue(fdStartTimeCouple_t fd_start_time_couple):	Συνάρτηση που υλοποιεί κάποιες λειτουργίες της ουράς.
fdStartTimeCoyple_t dequeue(void):	Συνάρτηση που υλοποιεί κάποιες λειτουργίες της ουράς.
bool is_empty(void):	Συνάρτηση που υλοποιεί κάποιες λειτουργίες της ουράς.
bool is_full(void):	Συνάρτηση που υλοποιεί κάποιες λειτουργίες της ουράς.
initialize_queue(void):	Συνάρτηση αρχικοποίησης.
initialize_global_variables(void):	Συνάρτηση αρχικοποίησης.
void sigint_hadler(int sig):	Χειριστής σήματος SIGINT.
void sigstp_hadler(int sig):	Χειριστής σήματος SIGSTP.

Όλες οι δομές και συναρτήσεις που έχουν να κάνουν με τον διακομιστή βρίσκονται στο αρχείο server.c ανάλογα όλες οι δομές και συναρτήσεις που έχουν να κάνουν με τον πελάτη βρίσκονται στο αρχείο client.c.

Μετρήσεις

Όνομα υπολογιστή	opti7020ws13
Επεξεργαστής	Intel i5-4590
Πλήθος πυρήνων	4
Μεταφραστής	gcc 4.8.4

Τα προγράμματα εκτελέστηκαν στο σύστημα που αναφέραμε στην εισαγωγή και η χρονομέτρηση έγινε με τη συνάρτηση gettimeofday(). Χρησιμοποιήθηκαν οι εξής αριθμοί νημάτων (1, 2, 3, 4, 8, 16), τα εξής μεγέθη ουράς (1, 4, 30) και 1032 αιτήσεις PUT και GET. Κάθε πείραμα εκτελέστηκε από τέσσερις φορές και υπολογίστηκαν οι μέσοι όροι. Οι χρόνοι είναι όλοι τους σε μονάδες second.

Πίνακας μετρήσεων χρόνου αναμονής με μέγεθος ουράς 1					
αριθμός νημάτων	1st run	2nd run	3rd run	4th run	Μέσος Χρόνος
1	0.000022	0.000027	0.000026	0.000023	0.0000245
2	0.000020	0.000020	0.000025	0.000020	0.00002125
3	0.000009	0.000010	0.000008	0.000008	0.00000875
4	0.000010	0.000008	0.000007	0.000010	0.00000875
8	0.000012	0.000008	0.000012	0.000009	0.00001025
16	0.000022	0.000020	0.000020	0.000020	0.0000205

Πίνακας μετρήσεων χρόνου εξυπηρέτησης με μέγεθος ουράς 1					
αριθμός νημάτων	1st run	2nd run	3rd run	4th run	Μέσος Χρόνος
1	0.000027	0.000029	0.000029	0.000028	0.00002825
2	0.000026	0.000026	0.000028	0.000026	0.0000265
3	0.000043	0.000038	0.000046	0.000044	0.00004275
4	0.000043	0.000039	0.000046	0.000042	0.0000425
8	0.000043	0.000041	0.000042	0.000044	0.0000425
16	0.000043	0.000044	0.000046	0.000044	0.00004425

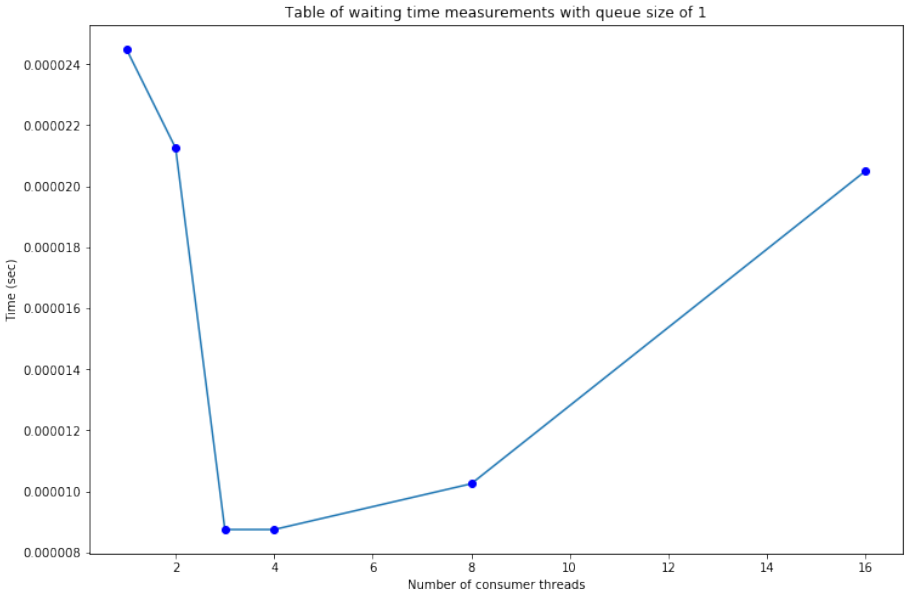


Table of waiting time measurements with queue size of 1
Time (sec)
0.000024
0.000022
0.000020
0.000018
0.000016
0.000014
0.000012
0.000010
0.000008
2
4
6
8
10
12
14
16
Number of consumer threads

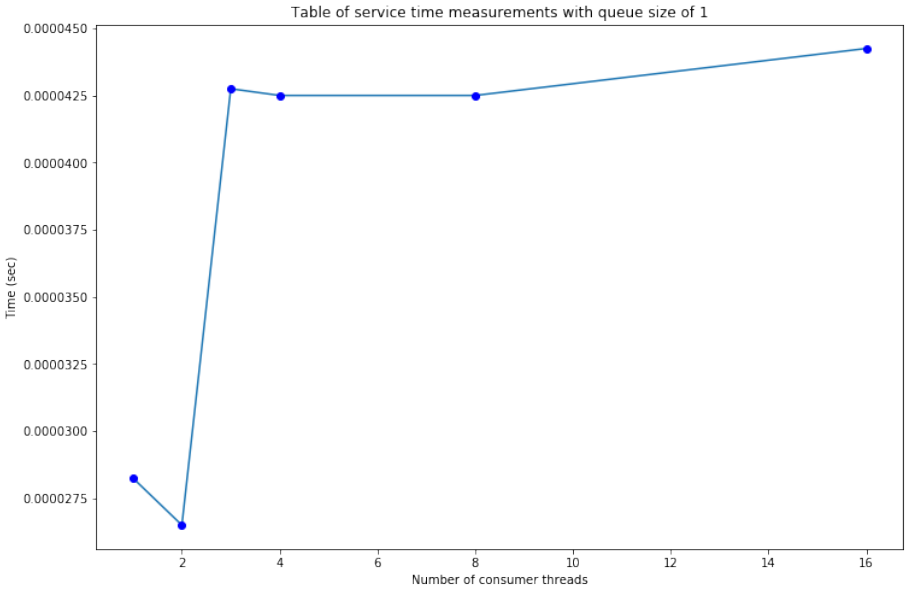
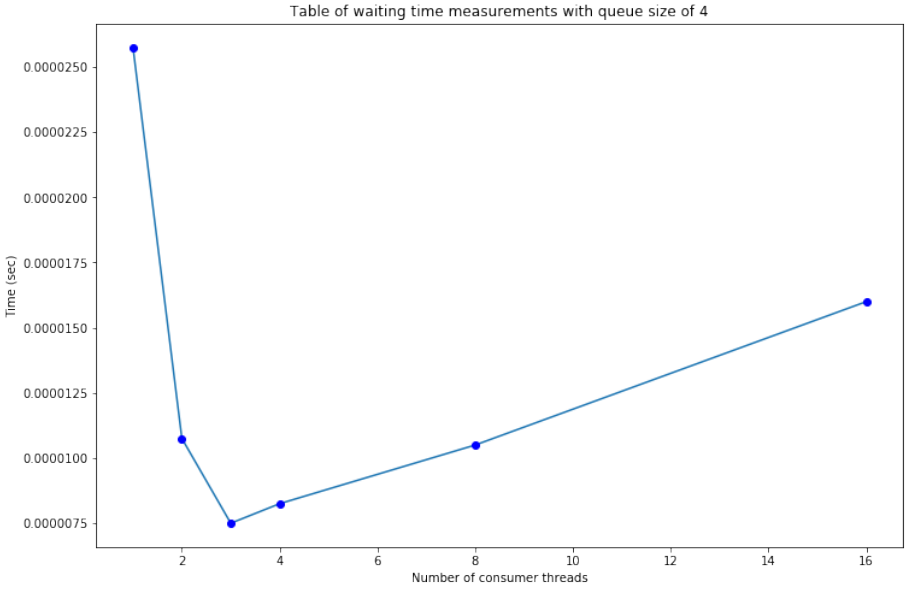
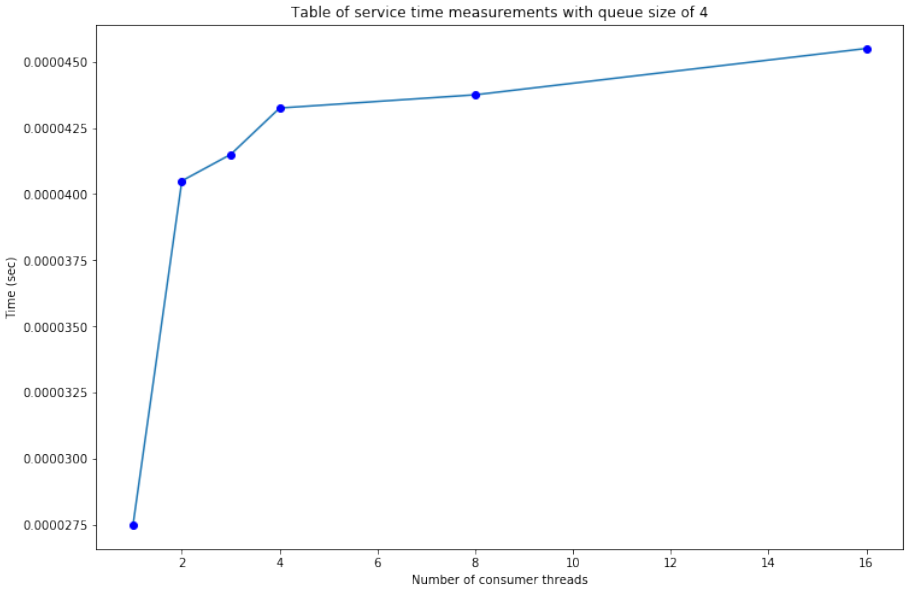


Table of service time measurements with queue size of 1
Time (sec)
0.0000450
0.0000425
0.0000400
0.0000375
0.0000350
0.0000325
0.0000300
0.0000275
2
4
6
8
10
12
14
16
Number of consumer threads

Πίνακας μετρήσεων χρόνου αναμονής με μέγεθος ουράς 4					
αριθμός νημάτων	1st run	2nd run	3rd run	4th run	Μέσος Χρόνος
1	0.000027	0.000026	0.000025	0.000025	0.00002575
2	0.000013	0.000010	0.000010	0.000010	0.00001075
3	0.000009	0.000007	0.000007	0.000007	0.0000075
4	0.000008	0.000008	0.000008	0.000009	0.00000825
8	0.000012	0.000009	0.000011	0.000010	0.0000105
16	0.000015	0.000016	0.000017	0.000016	0.000016

Πίνακας μετρήσεων χρόνου εξυπηρέτησης με μέγεθος ουράς 4					
αριθμός νημάτων	1st run	2nd run	3rd run	4th run	Μέσος Χρόνος
1	0.000027	0.000026	0.000029	0.000028	0.0000275
2	0.000039	0.000041	0.000040	0.000042	0.0000405
3	0.000038	0.000042	0.000043	0.000043	0.0000415
4	0.000043	0.000040	0.000046	0.000044	0.00004325
8	0.000042	0.000044	0.000045	0.000044	0.00004375
16	0.000042	0.000048	0.000049	0.000043	0.0000455





Πίνακας μετρήσεων χρόνου αναμονής με μέγεθος ουράς 30					
αριθμός νημάτων	1st run	2nd run	3rd run	4th run	Μέσος Χρόνος
1	0.000027	0.000024	0.000026	0.000025	0.0000255
2	0.000012	0.000012	0.000012	0.000013	0.00001225
3	0.000009	0.000009	0.000008	0.000007	0.0000825
4	0.000008	0.000007	0.000010	0.000007	0.000008
8	0.000007	0.000010	0.000008	0.000009	0.0000085
16	0.000009	0.000015	0.000015	0.000009	0.000012

Πίνακας μετρήσεων χρόνου εξυπηρέτησης με μέγεθος ουράς 30					
αριθμός νημάτων	1st run	2nd run	3rd run	4th run	Μέσος Χρόνος
1	0.000029	0.000027	0.000029	0.000028	0.00002825
2	0.000040	0.000039	0.000041	0.000038	0.0000395
3	0.000041	0.000045	0.000039	0.000048	0.00004325
4	0.000039	0.000042	0.000043	0.000043	0.00004175
8	0.000042	0.000041	0.000042	0.000044	0.00004225
16	0.000041	0.000043	0.000040	0.000041	0.00004125

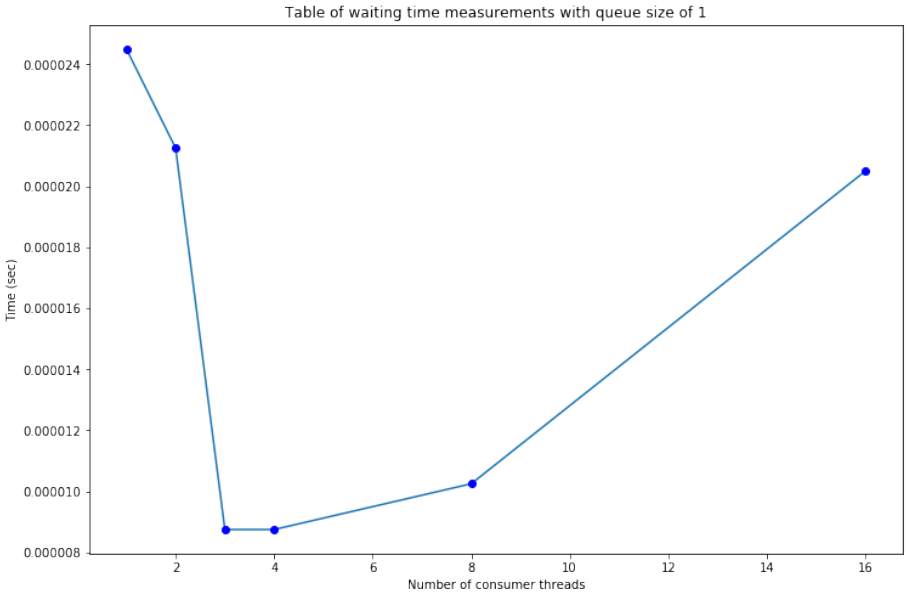


Table of waiting time measurements with queue size of 1
Time (sec)
0.000024
0.000022
0.000020
0.000018
0.000016
0.000014
0.000012
0.000010
0.000008
2
4
6
8
10
12
14
16
Number of consumer threads

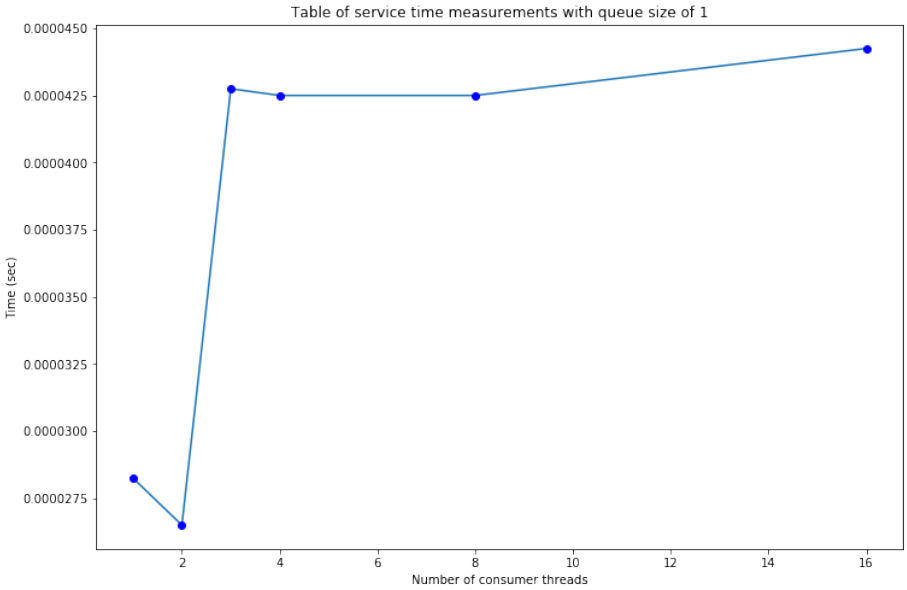
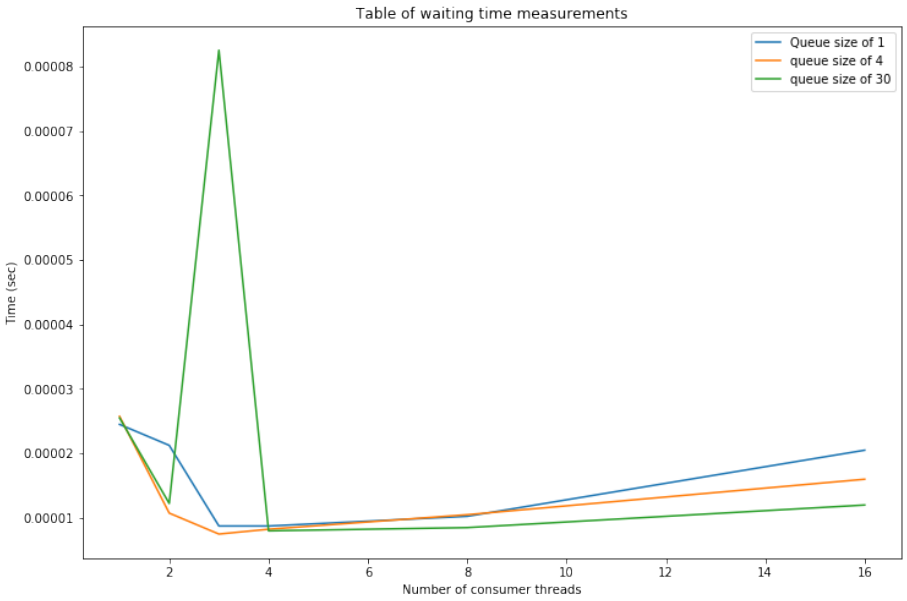
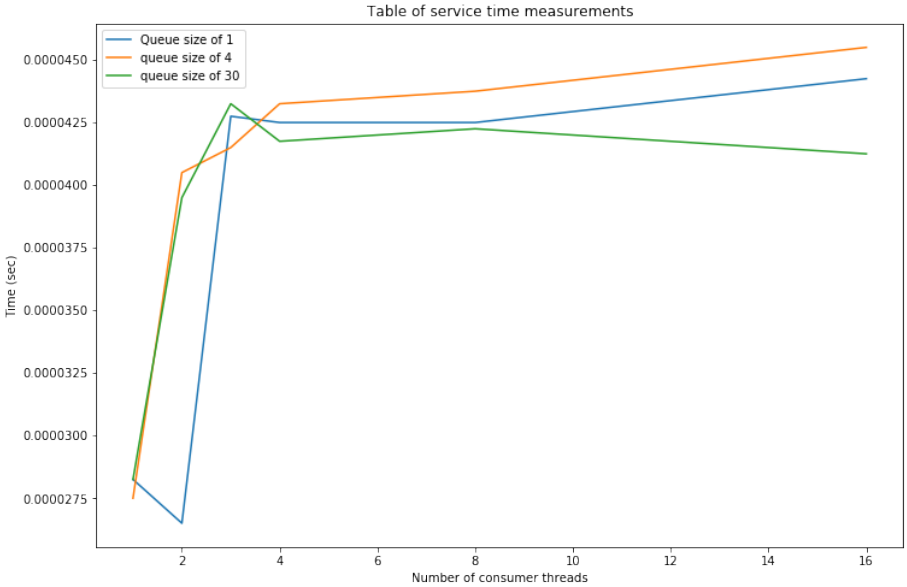


Table of service time measurements with queue size of 1
Time (sec)
0.0000450
0.0000425
0.0000400
0.0000375
0.0000350
0.0000325
0.0000300
0.0000275
2
4
6
8
10
12
14
16
Number of consumer threads

Παρατηρήσεις





Με βάση τα αποτελέσματα βλέπουμε ότι η αύξηση του μεγέθους της ουράς μειώνει τους χρόνους εκτέλεσης των νημάτων διακομιστή, αλλά η μείωση αυτή δεν είναι πολύ μεγάλη. Αντιθέτως βλέπουμε ότι ο αριθμός των νημάτων που δημιουργούμε κάνουν τις μεγαλύτερες διαφορές στους χρόνους εκτέλεσης. Πιο συγκεκριμένα όταν ο αριθμός των νημάτων είναι 2 ή 3 ή 4 ή 8 ο χρόνος αναμονής μειώνετε δραματικά, όχι τόσο με 16. Επομένως ο καλύτερος αριθμός νημάτων για

την μείωση του χρόνου αναμονής είναι 4 ή ο αριθμός των πυρήνων του επεξεργαστή. Όσο αφορά τους χρόνους εξυπηρέτησης των αιτήσεων παρατηρούμε ότι οι χρόνοι αυξάνονται ομαλά με την αύξηση των νημάτων.

Τα αποτελέσματα των μετρήσεων είναι λογικά και αναμενόμενα. Αρχικά με το μέγεθος της ουράς είναι λογικό να μειώνονται οι χρόνοι μίας και όσο πιο μεγάλη είναι τόσο μικρότερες οι πιθανότητες να αναγκάζονται τα νήματα να περιμένουν και να κλειδώνουν για πολύ χρόνο στις κλειδαριές του αμοιβαίου αποκλεισμού που ελέγχουν αν η ουρά είναι γεμάτη ή άδεια. Όσο αφορά την μείωση του χρόνου αναμονής, όσα περισσότερα τα νήματα τόσο πιο γρήγορα φεύγουν οι αιτήσεις από την ουρά αλλά μετά από έναν αριθμό νημάτων τα αποτελέσματα χειροτερεύουν γιατί δεν κάνει κάποια διαφορά να υπάρχουν πάρα πολλά νήματα και να μένουν αδρανή, το μόνο που καταφέρνουν είναι να τρώνε χρόνο κολλώντας στις κλειδαριές οι οποίες είναι γενικά χρονοβόρες κλήσεις. Γι αυτόν τον λόγο και ο καλύτερος αριθμός νημάτων που θα μπορούσαμε να χρησιμοποιήσουμε είναι ο αριθμός των επεξεργαστών, δηλαδή 4 στο συγκεκριμένο μηχάνημα. Τέλος η αύξηση του χρόνου εξυπηρέτησης με την αύξηση του αριθμό νημάτων μπορεί να εξηγηθεί με το φαινόμενο του κόκκου παραλληλισμού. Δηλαδή η εξυπηρέτηση των αιτήσεων δεν χρειάζεται πολύ χρόνο και η παραλληλοποίηση αυτό που στην πραγματικότητα καταφέρνει είναι μεν να μειώνει λίγο τον χρόνο εξυπηρέτησης του αλλά παράλληλα των αυξάνουν πολύ με τις κλείσεις των εντολών για την πραγματοποίηση του αμοιβαίου αποκλεισμού και τις μεταβλητές συνθήκες.

Ομάδα

Στην ομάδα είναι μόνο ένα άτομο (Σιρινιάν Αράμ Εμμανουήλ, 2537)