Ενσωματωμένα συστήματα πραγματικού χρόνου

Αναφορά – Εργασία 1

- Όνομα: Στασινός Αλκιβιάδης
- ΑΕΜ: 9214
- Link: https://github.com/astasinos/University-Projects/blob/master/Real-Time%20Embedded%20Systems/First%20Project/prod-cons.c

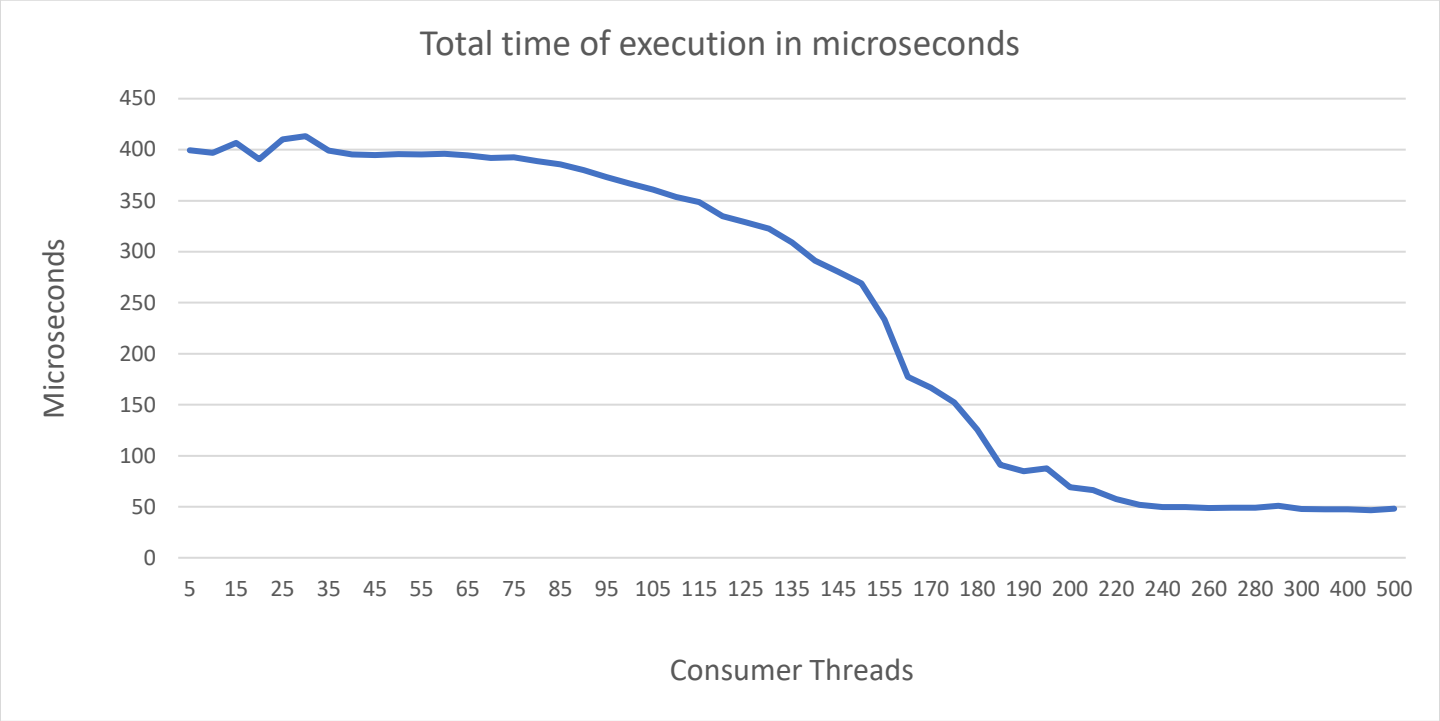
Όπως ζητείται και στην εκφώνηση ο κώδικας ο οποίος μας δόθηκε διαμορφώθηκε κατάλληλα ώστε η ***FIFO*** ουρά να δέχεται ***workFunc structs***, στα οποία προστέθηκε και ένα *timestamp* με τη μορφή ενός ***timeval struct*** όπου καταγράφεται η χρονική στιγμή κατά την οποία το *task* προστίθεται στην ουρά. Η εκτέλεση των συναρτήσεων που περιλαμβάνονται στα ***workFunc structs*** λαμβάνει χώρα **μετά** το **unlock** του **mutex** σε κάθε **consumer** με στόχο την παράλληλη εκτέλεση τους. Οι συναρτήσεις που υλοποιήθηκαν είναι απλές (τυπώνουν ένα μήνυμα ή το αποτέλεσμα μιας μαθηματικής πράξης), ενώ ως όρισμα δίνεται το id της εκάστοτε producer thread modulo 20000.

Για τον υπολογισμό της χρονικής διαφοράς χρησιμοποιήθηκε η συνάρτηση ***gettimeofday()***, ενώ για τη συλλογή των στατιστικών δεδομένων ακολουθήθηκε η εξής διαδικασία:

- Ο αριθμός των **producer threads** ορίστηκε σε **80.**
- Ο αριθμός **LOOP** τέθηκε **20000** με σκοπό την προσομοίωση συνθηκών υψηλού φόρτου.
- Προστέθηκε λειτουργία καταγραφής σε αρχείο του μέσου χρόνου που υπολογίζεται σε κάθε εκτέλεση από το πρόγραμμα.
- Ύστερα προς διευκόλυνση της συλλογής των δεδομένων δημιουργήθηκε ένα **python script** το οποίο αναλαμβάνει να εκτελέσει το πρόγραμμα για αρκετούς διαφορετικούς αριθμούς **consumer threads**. Για την απόκτηση ποιοτικότερων μετρήσεων το **script** εκτελεί **δύο** φορές το πρόγραμμα για κάθε αριθμό **consumer threads**, υπολογίζει τον μέσο όρο των δύο εκτελέσεων και καταγράφει την τελική μέτρηση. Υπάρχει στο repository της εργασίας.
- Δημιουργήθηκε το διάγραμμα των χρόνων μέσω ενός **Excel workbook** που υπάρχει στο repository.

Ο αριθμός των producer και των consumer δίνεται ως όρισμα κατά την εκτέλεση, για παράδειγμα ./prod-cons 80 200 (80 producer threads και 200 consumer).

Οι μετρήσεις έγιναν σε σύστημα με διπύρηνο επεξεργαστή **Intel Core i7 6500U** και φαίνονται παρακάτω.



Όπως φαίνεται ο χρόνος εκτέλεσης είναι σχετικά σταθερός αρχικά, και η πτώση ξεκινά περίπου στα 85 consumer threads. Από τις μετρήσεις προκύπτει πως ο βέλτιστος αριθμός από **consumers** στην περίπτωση μας είναι **260** κάτι που αντικατοπτρίζεται και στο παραπάνω διάγραμμα. Ο αριθμός αυτός επιλέχθηκε επειδή ελαχιστοποιεί το χρόνο εκτέλεσης και η περαιτέρω αύξηση των **threads** δε δείχνει σημαντική βελτίωση.