Ενσωματωμένα Συστήματα Πραγματικού Χρόνου

Εργασία 2η: Producer Consumer Implementation with Timer

Author

Όνομα: Πορτοκαλίδης Σταύρος, ΑΕΜ: 9334, email: stavport@ece.auth.gr

Code: https://github.com/sportokalidis/producer-consumer-pi

Περιγραφή Προβλήματος

Σκοπός της 2ης εργασίας είναι η υλοποίηση μιας ουράς εξυπηρέτησης, με χρήση πολλαπλών threads στο ρόλο των producers και consumers όπως είχε γίνει και στα πλαίσια της πρώτης εργασίας. Η διαφορά έγκειται στο γεγονός ότι στην 2η εργασία καλούμαστε να υλοποιήσουμε ένα timer ο οποίος με μια συγκεκριμένη περίοδο, θα καλείται και θα προσθέτει διεργασίες στην ουρά. Τέλος, καλούμαστε να αντιμετωπίσουμε και το πρόβλημα της χρονικής μετατόπισης (time drifting), ώστε να πραγματοποιείται εισαγωγή των διεργασιών σύμφωνα με την περίοδο που έχει οριστεί.

Εξήγηση Υλοποίησης

Για την υλοποίηση του timer δημιουργούμε ένα struct με όνομα Timer το οποίο περιέχει όλες τις απαραίτητες λειτουργίες όπως ζητείται από την εκφώνηση (period, taskToExecute, StartDelay, κτλ). Αρχικά, στην main() function θεωρώντας ως δεδομένα τον χρόνο εκτέλεσης και την περίοδο του Timer υπολογίζουμε τον αριθμό των task που θα κληθούν να εξυπηρετήσουν οι consumers. Στην συνέχεια, δημιουργούμε όλα τα consumers threads πρώτα (θα περιμένουν στο wait, εφόσον οι ούρα είναι άδεια), και έπειτα δημιουργούμε ένα αντικείμενο timer, στο οποίο αφού περάσουμε όλα τα απαραίτητα δεδομένα, μέσω της functions start(timer T), δημιουργούμε όλα τα producer threads. Η ρουτίνα που τρέχουν τα threads consumers είναι πανομοιότυπη με αυτήν της 1ης εργασίας, ωστόσο η ρουτίνα που εκτελούν τα producer threads έχει ορισμένες διαφοροποιήσεις για να προσαρμοστούμε στο ζητούμενο της 2ης εργασίας. Η κύρια διαφοροποίηση που υπάρχει είναι ότι τα threads στο τέλος της ρουτίνας κάνουν ένα sleep(). Ο χρόνος τον οποίο κάνει sleep το thread producer δεν είναι σταθερός. Ουσιαστικά, μετράμε τον χρόνο από την στιγμή που ξύπνησε από το προηγούμενο sleep, μέχρι και την στιγμή που ξανά έκανε insert στην ουρά (drift time). Έτσι, για να διατηρήσουμε σταθερή την περίοδο που μπαίνουν οι διεργασίες στην ουρά, βάζουμε τον producer να κάνει sleep για *period – drift time*.

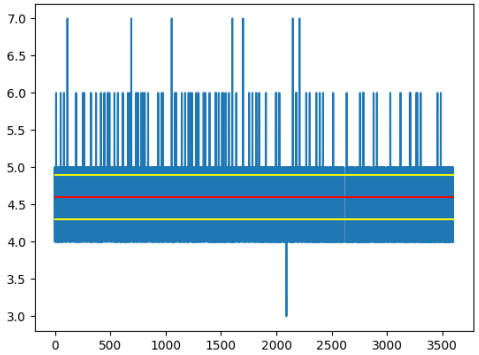
Testing

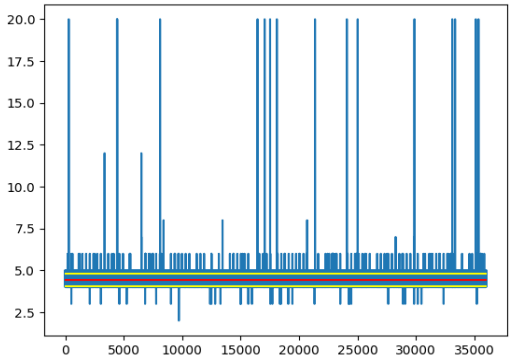
Ως επαλήθευση για την ορθότητα της υλοποίησης έχουμε το γεγονός ότι το πρόγραμμα τερματίζει (δηλαδή κάνουν return όλα τα threads) και οι ουρά είναι empty. Αυτό σε συνδυασμό ότι ο αριθμός των αποτελεσμάτων είναι ο αναμενόμενος και ότι εκτέλεση διαρκεί για την αναμενόμενη χρονική διάρκεια (μια ώρα), μας υποδηλώνει ότι εξυπηρετούνται όλες οι διεργασίες και με τον σωστό ρυθμό εισαγωγής στην ουρά.

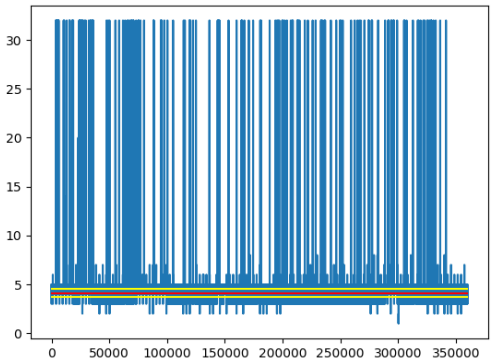
Μετρήσεις και Αποτελέσματα

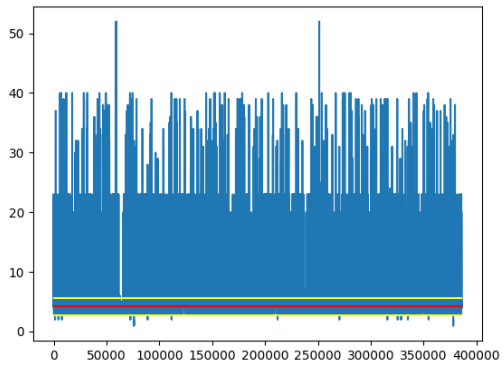
Για εκτέλεση και την συλλογή των μετρήσεων και των αποτελεσμάτων έγινε χρήση του Raspberry Pi 4. Για τις παρακάτω μετρήσεις έχουν επιλεγεί QueueSize = 5, πλήθος Consumers Q = 2 και πλήθος Producers P = 1. Τα στατιστικά που παρουσιάζονται, είναι για εκτέλεση του κώδικα για χρονική διάρκεια μιας ώρας. Μελετάμε τους χρόνους που πήραμε για το time drifting, την χρονική διάρκεια που ο Producer χρειάστηκε για να εισέρθει στην ουρά (Producer Waiting Time) και τον χρόνο που η διεργασία περίμενε στην ουρά μέχρι να ξεκινήσει η εκτέλεση της (Time in Queue). Το κάθε γράφημα που παρουσιάζεται είναι για μια διαφορετική περίοδο του timer (1s, 0.1s, 0.01s, all). Στον οριζόντιο άξονα έχουμε το πλήθος των διεργασιών και στον κατακόρυφο την χρονική διάρκεια σε useconds.

- Drift Time



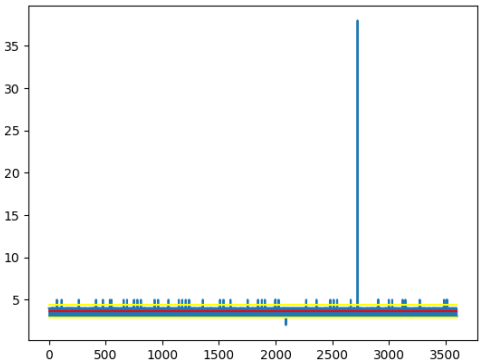


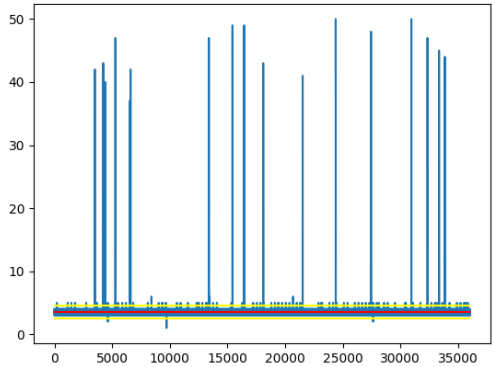


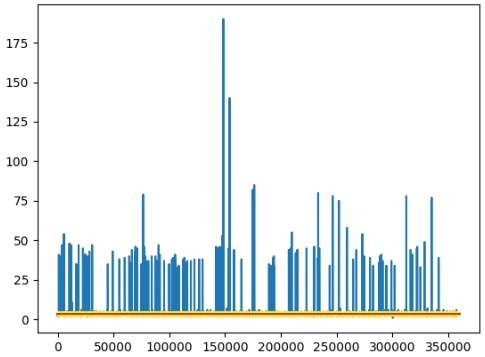


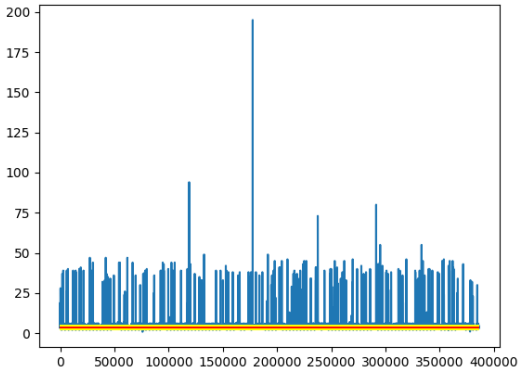
Period	Mean	Median	Min	Max	Std
1	4.6	5	3	7	0.29
0.1	4.4	4	2	20	0.36
0.01	4.11	4	1	32	0.43
All	4.20	4	1	52	1.48

- Producer Waiting Time



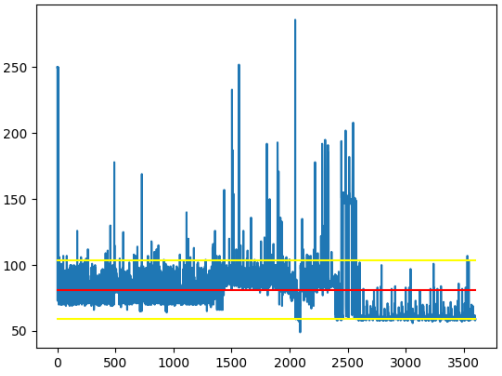


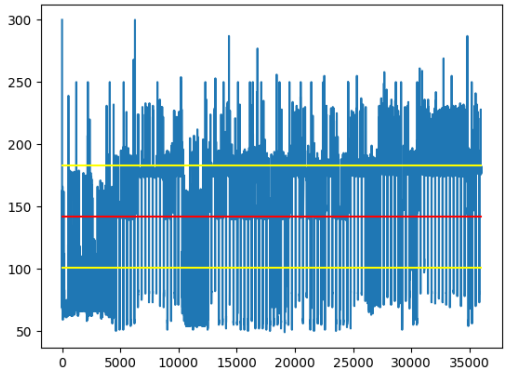


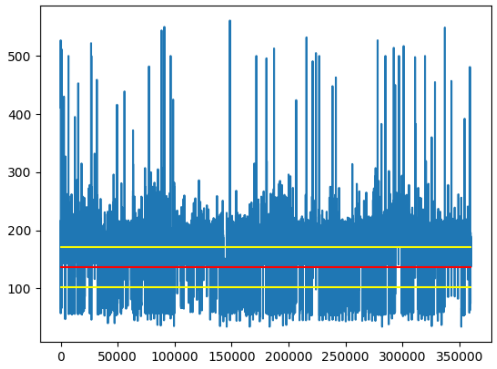


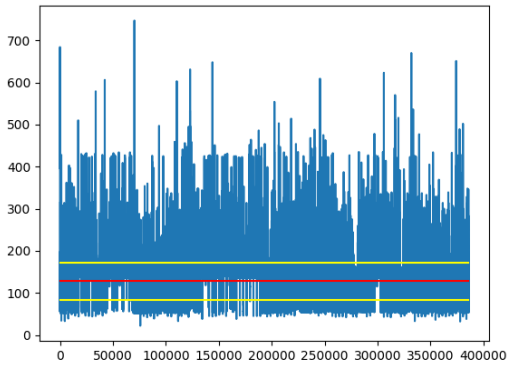
Period	Mean	Median	Min	Max	Std
1	81.3	81	49	289	22.5
0.1	141.7	147	49	300	41.1
0.01	136.9	144	34	561	34.1
All	128.1	144	22	747	44.2

- Time in Queue









Period	Mean	Median	Min	Max	Std
1	3.6	4	2	38	0.76
0.1	3.5	4	1	50	1.03
0.01	3.5	4	1	190	0.98
All	1	4	1	195	0.99

- CPU Usage

	User Time	System Time	Real Time	CPU Usage
T=1	0m0.736s	0m2.432s	60m2.325s	$8x10^{-4}$ %
T=0.1	0m1.536s	0m5.856s	60m9.314s	$2x10^{-3}$ %
T=0.01	0m14.287s	1m1.044s	60m12.156s	0.021 %
All	0m14.973s	1m9.346s	60m14.843s	0.023 %

Σχόλια και Συμπεράσματα

Από τα αποτελέσματα και την στατιστική ανάλυση διαπιστώνουμε ότι η παρότι υπάρχουν ακραίες τιμές στις μετρήσεις μας, το μεγαλύτερο πλήθος των διεργασιών παρουσιάζει κοντινά αποτελέσματα σε χρόνους, όπως φαίνεται από την τυπική απόκλιση στα γραφήματα που δεν απέχει κατά πολύ από τον μέσο όρο. Επιπλέον αξίζει να σημειωθεί ότι ο χρόνος που παραμένει μια διεργασία στην ουρά, μπορεί να μειωθεί με την χρήση περισσότερων consumer threads, όπως επίσης, ο χρόνος που χρειάζεται ένα producer thread για να εισέρθει στην ουρά μειώνεται με την επιλογή μιας μεγαλύτερης ουράς. Τέλος, όπως παρατηρούμε και από τις μετρήσεις για το CPU Usage, η υλοποίηση έτρεχε όντως για χρονική διάρκεια μιας ώρας και τα αποτελέσματα που πήραμε είχαν το σωστό πλήθος.