Ενσωματωμένα Συστήματα Πραγματικού Χρόνου

Εργασία 1

Σύνδεσμος κώδικα: Github

Όνομα : Αλέξανδρος Δελητζάς
ΑΕΜ : 8448
e-mail : adelitzas@ece.auth.gr

9 Μαΐου 2019

1 Εισαγωγή

Στα πλαίσια της εργασίας αναπτύχθηκαν δύο προγράμματα με σκοπό να γίνει τακτική δειγματοληψία με όσο το δυνατό μικρότερη απόκλιση από τον πραγματικό χρόνο. Στην πρώτη υλοποίηση γίνεται χρήση της nanosleep(). Για αυτήν την περίπτωση, εξηγείται η ανάγκη για χρήση των timestamps που συλλέγονται με σκοπό τη βελτίωση της ακρίβειας του χρόνου δειγματοληψίας. Στη δεύτερη υλοποίηση γίνεται χρήση της setitimer() και διακοπών (interrupts). Σε αυτή την περίπτωση, η χρήση των timestamps που συλλέγονται για βελτίωση δεν είναι απαραίτητη. Για την ανάλυση της στατιστικής συμπεριφοράς, τρέξαμε την καθε υλοποίηση για συνολική διάρκεια $t = 7200\,sec$ και περίοδο δειγματοληψίας $\delta t = 0.1\,sec$. Τέλος, θα εξετάσουμε την κατανάλωση ενέργειας των δύο υλοποιήσεων.

2 Πρώτη Υλοποίηση

2.1 Περιγραφή υλοποίησης

Σε πρώτη φάση, με σκοπό να ρυθμίσουμε τις παραμέτρους της nanosleep(), εκφράζουμε το χρονικό διάστημα δt σε ακέραια seconds και nanoseconds. Ο βασικός πυρήνας του προγράμματος είναι ένας βρόχος επανάληψης, ο οποίος τερματίζει όταν συλλεχθεί ο απαραίτητος αριθμός δειγμάτων. Σε κάθε επανάληψη η διεργασία παραμένει αδρανής για το χρονικό διάστημα που ορίζουμε και μετά καλείται η συνάρτηση timerHandler() για να συλλέξει το τρέχον timestamp. Τα timestamps αποθηκεύονται σε έναν πίνακα. Μόλις περάσει ο συνολικός χρόνος δειγματοληψίας ή η διαδικασία διακοπεί από το χρήστη μέσω Ctrl-C, τότε αποθηκεύονται σε ένα αρχείο.
Σε μια πρώτη απόπειρα, ορίσαμε το χρονικό διάστημα της nanosleep() ίσο με δt σε κάθε επανάληψη. Όμως, παρατηρήθηκε ότι συσσωρευόταν σφάλμα από κάθε επανάληψη, με αποτέλεσμα καθώς αυξάνονται οι επαναλήψεις να υπάρχει μεγάλη απόκλιση από τον ιδανικό χρόνο και να χάνονται δείγματα.

Για να βελτιώσουμε την ακρίβεια των χρονικών στιγμών δειγματοληψίας, χρησιμοποιήσαμε τα timestamps που συλλέγουμε. Έτσι, κάθε φορά που καλείται η timerHandler() προσαρμόζει το επόμενο χρονικό διάστημα της nanosleep() με τον εξής τρόπο:

$$\delta t_{next} = \delta t - dif, \qquad dif = t_{real} - t_{ideal}$$

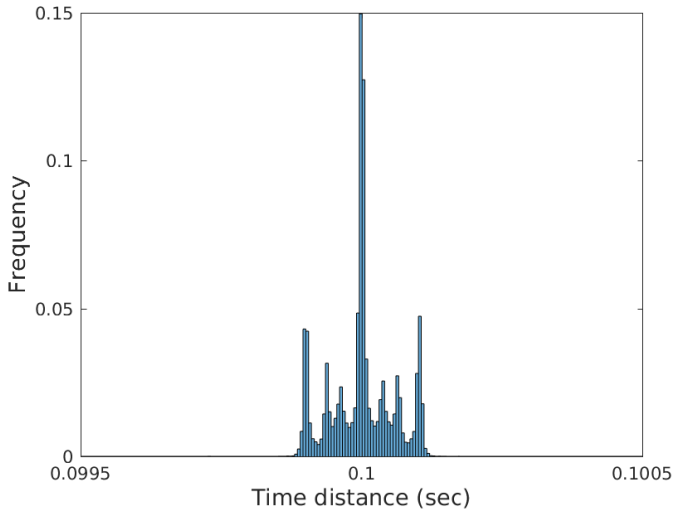
όπου dif είναι η διαφορά του timestamp που συλλέξαμε από το ιδανικό. Στην πραγματικότητα αντί για dif στην υλοποίηση μας, παίρνουμε το floating-point υπόλοιπο της διαίρεσης $dif/\delta t$ ώστε σε περίπτωση που χαθεί κάποιο timestamp να μην επηρεαστεί ο χρόνος που θα ληφθεί το επόμενο.

Μέγιστο (sec)	0.108572
Ελάχιστο (sec)	0.091453
Μέσος όρος (sec)	0.100000
Διάμεσος (sec)	0.100000
Τυπική απόκλιση (sec)	0.000078

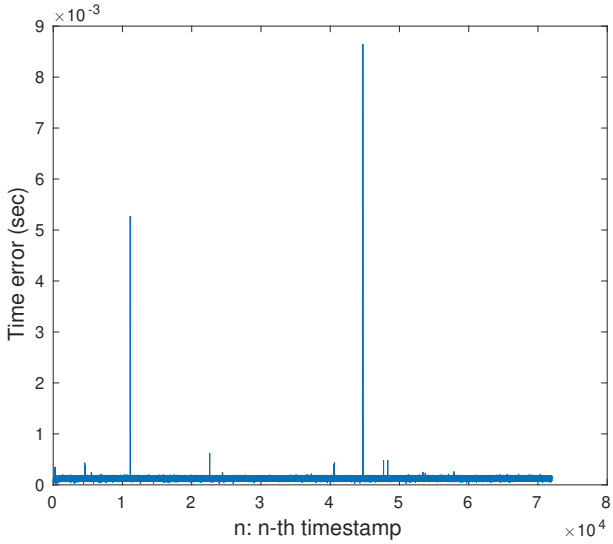
Πίνακας 1: Στατιστικά στοιχεία της χρονικής απόστασης μεταξύ δύο διαδοχικών δειγμάτων

2.2 Στατιστική συμπεριφορά

Από το πείραμα που έγινε, τα στατιστικά στοιχεία της χρονικής απόστασης μεταξύ δύο διαδοχικών δειγμάτων φαίνονται στον Πίνακα 1. Αρχικά, ο αριθμός των δειγμάτων που λάβαμε είναι σωστός (το τελευταίο δείγμα λήφθηκε τη χρονική στιγμή $t = 7200.000077\,sec$). Όπως βλέπουμε, ο μέσος όρος και η διάμεσος είναι ίσες με το $\delta t = 0.1$ που χρησιμοποιήσαμε το οποίο είναι επιθυμητό στοιχείο. Η τυπική απόκλιση είναι της τάξεως των $10^{-5}\,sec$. Από τα στοιχεία αυτά συμπεραίνουμε ότι οι χρονικές αποστάσεις μεταξύ διαδοχικών δειγμάτων



(α′) Κατανομή των χρονικών αποστάσεων μεταξύ δύο διαδοχικών διεγμάτων



(β′) Απόκλιση των χρονικών στιγμών που λήφθηκαν τα timestamps από τις ιδανικές

δεν απομακρύνονται πολύ από το $\delta t = 0.1$.
Στο Σχήμα 1α′ βλέπουμε την κατανομή των χρονικών αποστάσεων μεταξύ δύο διαδοχικών δειγμάτων. Όπως περιμέναμε, η πλειοψηφία των χρονικών αποστάσεων λαμβάνουν τιμές πολύ κοντά στο $\delta t = 0.1$. Έπειτα όσο απομακρυνόμαστε από το 0.1 η συχνότητα ακολουθεί μια καθοδική πορεία. Εξαίρεση αποτελούν κάποια μεμονωμένα bins, στα οποία το γράφημα θα ήταν πιθανότατα πιο εξομαλυμένο αν το πείραμα συνεχιζόταν για περισσότερο από $t = 7200\,sec$. Το εύρος που βρίσκονται οι μη αμελητέες συχνότητες στην κατανομή είναι $2 \cdot 10^{-4}\,sec$.
Στο Σχήμα 1β′ βλέπουμε την απόκλιση των χρονικών στιγμών που λήφθηκαν τα timestamps από τις ιδανικές. Το σφάλμα αυτό είναι της τάξεως $10^{-4}\,sec$ και σε κάποιες περιπτώσεις λαμβάνει πολύ μικρές τιμές τάξεως $10^{-3}\,sec$.

3 Δεύτερη Υλοποίηση

3.1 Περιγραφή υλοποίησης

Αρχικά, με σκοπό να ρυθμίσουμε τις παραμέτρους της setitimer(), εκφράζουμε το χρονικό διάστημα δt σε ακέραια seconds και microseconds. Χρησιμοποιήσαμε την παράμετρο ITIMER_REAL της setitimer(), ώστε να μετρήσουμε σε διαστήματα πραγματικού χρόνου. Έτσι, κάθε φορά που περνάει χρονικό διάστημα δt θα σταλεί σήμα SIGALRM στη διεργασία.
Και σε αυτήν την υλοποίηση, ο βασικός πυρήνας του προγράμματος είναι ένας βρόχος επανάληψης, ο οποίος τερματίζει όταν συλλεχθεί ο απαραίτητος αριθμός δειγμάτων. Λόγω της pause(), σε κάθε επανάληψη η διεργασία παραμένει αδρανής μέχρι να σταλεί το σήμα SIGALRM. Μόλις συμβεί αυτό, αναλαμβάνει δράση η timerHandler() (handler του σήματος SIGALRM), με σκοπό να συλλέξει το timestamp.

3.2 Στατιστική συμπεριφορά

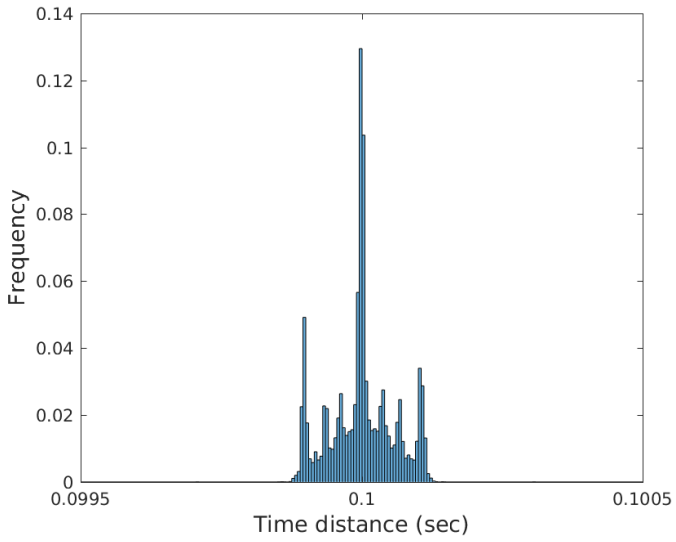
Από το πείραμα που έγινε, τα στατιστικά στοιχεία της χρονικής απόστασης μεταξύ δύο διαδοχικών δειγμάτων φαίνονται στον Πίνακα 2. Και πάλι ο αριθμός των δειγμάτων που λάβαμε είναι σωστός (το τελευταίο δείγμα λήφθηκε τη χρονική στιγμή $t = 7200.000019\,sec$). Όπως

και στην προηγούμενη υλοποίηση, ο μέσος όρος και η διάμεσος είναι (σχεδόν) ίσες με το $\delta t = 0.1$. Η τυπική απόκλιση, όμως, λαμβάνει τιμή λίγο μικρότερη. Επίσης, η ελάχιστη και η μέγιστη τιμή βρίσκονται πιο κοντά στο $\delta t = 0.1$, από ότι πριν.

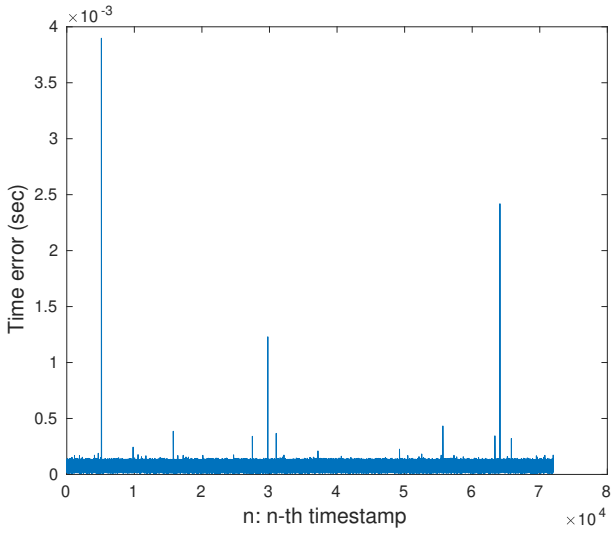
Στο Σχήμα 2α΄ παρατηρούμε ότι και σε αυτή την υλοποίηση η πλειοψηφία των αποστάσεων βρίσκεται γύρω από το $\delta t = 0.1$, όπως είναι το επιθυμητό. Η διαφορά με την προηγούμενη υλοποίηση είναι ότι κάποια μεμονωμένα bins που χαλούσαν την καθοδική πορεία της κατανομής, καθώς απομακρυνόμαστε από το κέντρο ($\delta t = 0.1$), πλέον έχουν μικρότερη συχνότητα.
Όσον αφορά την απόκλιση από τις ιδανικές χρονικές στιγμές δειγματοληψίας (Σχήμα 2β΄) και πάλι είναι της τάξεως 10^{-4} με 10^{-3} και είναι ελαφρώς μικρότερη από της προηγούμενης υλοποίησης.
Συνοπτικά, οι δύο υλοποιήσεις δεν έχουν μεγάλες διαφορές μεταξύ τους. Όμως, η δεύτερη υλοποίηση αποδείχτηκε λίγο πιο αξιόπιστη.

Μέγιστο (sec)	0.103765
Ελάχιστο (sec)	0.096236
Μέσος όρος (sec)	0.100000
Διάμεσος (sec)	0.099999
Τυπική απόκλιση (sec)	0.000063

Πίνακας 2: Στατιστικά στοιχεία της χρονικής απόστασης μεταξύ δύο διαδοχικών δειγμάτων



(α΄) Κατανομή των χρονικών αποστάσεων μεταξύ δύο διαδοχικών διεγμάτων



(β΄) Απόκλιση των χρονικών στιγμών που λήφθηκαν τα timestamps από τις ιδανικές

4 Κατανάλωση ενέργειας

Όπως αναφέραμε, και στις δύο υλοποιήσεις υπήρξε μέριμνα ώστε: • η διεργασία να παραμένει αδρανής κατά το χρονικό διάστημα αναμονής, • μετά το πέρας του διαστήματος να ΄ξυπνάει΄ για την καταγραφή του timestamp, • να επαναλαμβάνεται αυτή η διαδικασία σε όλη τη διάρκεια δειγματοληψίας.
Με σκοπό την παρακολούθηση της χρήσης της CPU από τις δύο υλοποιήσεις, δημιουργήσαμε ένα bash script (monitor_cpu.sh). Το script αυτό παίρνει επαναληπτικά μετρήσεις του ποσοστού χρήσης της CPU που κάνει κάποια διεργασία μέσω της εντολής ps. Κατά τη διάρκεια των πειραμάτων, θέσαμε την περίοδο μεταξύ των διαδοχικών μετρήσεων υποπολλαπλάσια του δt και τρέξαμε το script. Όπως διαπιστώθηκε, και οι δύο υλοποιήσεις κάναν μηδενική χρήση της CPU.