Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Ενσωματωμένα Συστήματα Πραγματικού Χρόνου 2024
Τελική Εργασία

Γαλάνη Δήμητρα
ΑΕΜ 10331

Μάθημα: Ενσωματωμένα Συστήματα Πραγματικού Χρόνου
Διδάσκων: Πιτσιάνης Νικόλαος

10 Μαρτίου 2025

Εισαγωγή

Στην παρούσα εργασία υλοποιείται ένας client, σε γλώσσα προγραμματισμού C, ο οποίος συνδέεται με τον server της ιστοσελίδας `https://finnhub.io/` με σκοπό την λήψη πακέτων μέσω websocket που αφορούν κάποια συναλλάγματα, την αποθήκευση τους και την εύρεση κάποιων στατιστικών μεγεθών σε πραγματικό χρόνο. Στην υλοποίηση έχει χρησιμοποιηθεί η βιβλιοθήκη libwebsockets, το cross-compile γίνεται στον υπολογιστή και το εκτελέσιμο αρχείο που προκύπτει μεταφέρεται μέσω ssh στο Raspberry Pi Zero 2 W και εκτελείται εκεί.

Υλοποίηση

Η υλοποίηση βασίζεται σε τρεις κυκλικές ουρές. Αρχικά έχει δημιουργηθεί ένα buffer σταθερού μεγέθους, το οποίο να είναι αρκετά μεγάλο προκειμένου να μπορεί να αποθηκεύσει τα πακέτα που έρχονται μέσα σε ένα λεπτό. Παρόλο που είναι αρκετά μεγάλο για να περιέχει τα πακέτα που έρχονται δεν πιάνει πολύ χωρό σε σχέση με τον συνολικό χώρο του raspberry. Το buffer αυτό αντιπροσωπεύει την RAM, στην οποία αποθηκεύονται τα δεδομένα τύπου Data, μια δομή στην οποία αποθηκεύονται η τιμή, το όνομα του συμβόλου, το timestamp της συναλλαγής και ο όγκος, και έχει υλοποιηθεί σαν κυκλική ουρά για οποία ισχύει ότι όταν είναι γεμάτη τα παλιότερα δεδομένα γίνονται overwrite. Στην ουρά WorkQueue, η οποία είναι FIFO, τοποθετούνται οι διεργασίες με τα ορίσματα τους που πρόκειται να εκτελεσθούν από τους consumers και κάθε φορά που είναι διαθέσιμος ένας consumer κάνει dequeue την διεργασία που βρίσκεται στο head της ουράς. Οι διεργασίες που τοποθετούνται είναι είτε διεργασίες για τον υπολογισμό των candlesticks είτε για τον υπολογισμό των delays. Τέλος, όσον αφορά την ουρά movingAverageQueue χρησιμοποιείται για τον υπολογισμό του moving average και του συνολικού όγκου των 15 λεπτών. Είναι και αυτή μια κυκλική ουρά που κάνει δηλαδή wrap around και όταν είναι γεμάτη σε κάθε λεπτό αποθηκεύεται το πιο πρόσφατο moving average και ο πιο πρόσφατος συνολικός όγκος στην θέση που ήταν τα παλιότερα στοιχεία. Οπότε τελικά σε κάθε λεπτό στην δομή αυτή βρίσκονται τα δεδομένα από τα 15 πιο πρόσφατα λεπτά και μπορεί εύκολα να υπολογιστεί το moving average και ο συνολικός όγκος με βάση τα 15 τελευταία λεπτά. Υπάρχουν ουσιαστικά τέσσερις κατηγορίες threads που εκτελούν κάτι διαφορετικό.

1. producer: ο οποίος λαμβάνει τα πακέτα μέσω του websocket τα κάνει parse τα JSON πακέτα και τα καταγράφει στα κατάλληλα αρχεία, εφόσον αναφέρεται στην εκφώνηση ότι πρέπει να γίνει η καταγραφή των συναλλαγών σε αρχεία τη στιγμή που λαμβάνεται τη πληροφορία.

2. timer: χρησιμοποιείται για να σταματήσει το πρόγραμμα αυτόματα μετά το πέρας των 48 ωρών καθώς κάνει sleep για ένα χρονικό διάστημα που ορίζεται και μετά θέτει την μεταβλητή activeClient σε 0 προκειμένου να κλείσει ο client.

3. scheduler: ενεργοποιείται κάθε λεπτό και βρίσκει το tail του buffer. Δηλαδή βρίσκει μέχρι αυτήν την χρονική στιγμή στο λεπτό το τελευταίο στοιχείο που τοποθετήθηκε στο buffer. Το head για το πρώτο λεπτό έχει οριστεί ως το head του buffer, ενώ για τα επόμενα λεπτά στο τέλος τις διεργασίας του scheduler τίθεται ως το επόμενο στοιχείο του tail, προκειμένου να μην έχουμε απώλεια δεδομένων. Οπότε σε κάθε λεπτό έχουμε δύο indices όπου ανάμεσα τους υπάρχουν τα δεδομένα που λήφθηκαν μέσα στο λεπτό. Προκειμένου να μην έχουμε μεγάλο drift, δηλαδή την χρονική μετατόπιση που προκύπτει στην κλήση της calculations, αντί να υπολογιστούν σε αυτήν την διεργασία τα candlesticks απλώς τοποθετείται η διεργασία calculations στην workQueue με ορίσματα τα δύο indices και τον χρόνο που γίνεται αυτό για τον υπολογισμό των καθυστερήσεων.

4. consumer: αφαιρεί από την WorkQueue την πρώτη διεργασία με τα ορίσματα της και την εκτελεί. Οι διεργασιές που τοποθετούνται στην ουρά είναι είτε η calculation είτε η tradeDelayCalculation. Έχει επιλεγεί να χρησιμοποιηθούν δύο consumers αφού η calculation τοποθετείται κάθε ένα λεπτό και η tradeDelayCalculation εκτελείται πολύ γρήγορα οπότε κρίθηκε ότι δεν θα υπάρχει μεγάλη αναμονή στην ουρά.

Η υλοποίηση χρησιμοποιεί κατά βάση την μεθολογία producer-consumer, καθώς ο producer τοποθετεί τα δεδομένα σε μία ουρά και οι consumers παίρνει, μέσω του scheduler, τα δεδομένα αυτά και πραγμα-

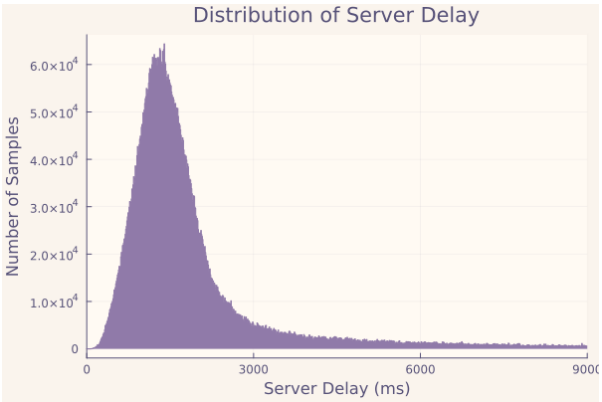
τοποιεί τους υπολογισμούς που του ανατίθενται. Περισσότερες λεπτομέρειες για τις δομές που χρησιμοποιούνται και για τις συναρτήσεις μπορούν να βρεθούν στα σχόλια του κώδικα.

Καθυστερήσεις

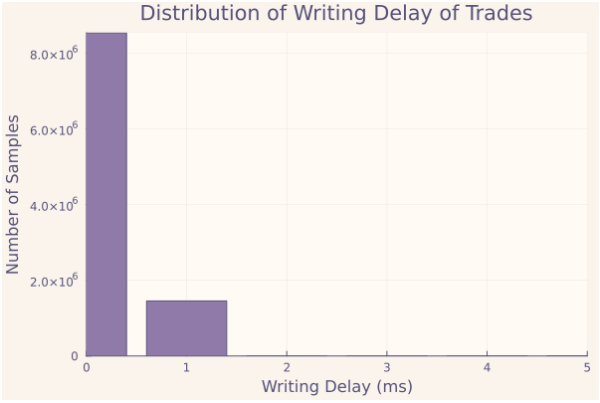
Στο πρόγραμμα υπολογίζονται οι εξής χρόνοι και καθυστερήσεις:

1. Καθυστέρηση του server: διαφορά της χρονικής στιγμής της λήψης της πληροφορίας με την χρονική στιγμή που είναι καταγεγραμμένη στο πακέτο στο χωρίο t, δηλαδή την χρονική στιγμή που πραγματοποιήθηκε η συναλλαγή.
2. Καθυστέρηση καταγραφής της συναλλαγής: διαφορά της χρονικής στιγμής της καταγραφής της συναλλαγής στο κατάλληλο αρχείο από την χρονική στιγμή της λήψης της πληροφορίας.
3. Καθυστέρηση υπολογισμού και καταγραφής του candlestick: διαφορά της χρονικής στιγμής του πέρατος του λεπτού για το οποίο πρέπει να υπολογιστεί το candlestick και της χρονικής στιγμής της ολοκλήρωσης του υπολογισμού και της καταγραφής του candlestick.
4. Καθυστέρηση υπολογισμού και καταγραφής του moving average: διαφορά της χρονικής στιγμής του πέρατος του λεπτού για το οποίο πρέπει να υπολογιστεί το moving average και της χρονικής στιγμής της ολοκλήρωσης του υπολογισμού και της καταγραφής του moving average.
5. Μετατόπιση κλήσης της συνάρτησης για τον υπολογισμό στο λεπτό: διαφορά της χρονικής στιγμής κατά την οποία θα έπρεπε να κληθεί η συνάρτηση, δηλαδή ακριβώς σε κάθε 60 δευτερόλεπτα, και την στιγμή που καλείται κάθε φορά η συνάρτηση για τον υπολογισμό. Υπάρχει μια μικρή μετατόπιση καθώς μέχρι να εκτελεστεί ξανά η εντολή sleep πρέπει να προστεθεί η διεγρασία calculation στην workQueue μαζί με τα ορίσματα της ενεργοποιώντας το mutex της ουράς. Η μετατόπιση αυτή θα μπορούσε να αντιμετωπιστεί, προβλέποντας την και ρυθμίζοντας κατάλληλα το χρονικό διάστημα της sleep, το οποίο στην συγκεκριμένη υλοποίηση είναι σταθερό.

Όλες οι καθυστερήσεις έχουν υπολογιστεί στην τάξη των msec. Παρακάτω παρέχονται κάποια διαγράμματα που απεικονίζουν τις καθυστερήσεις αυτές.



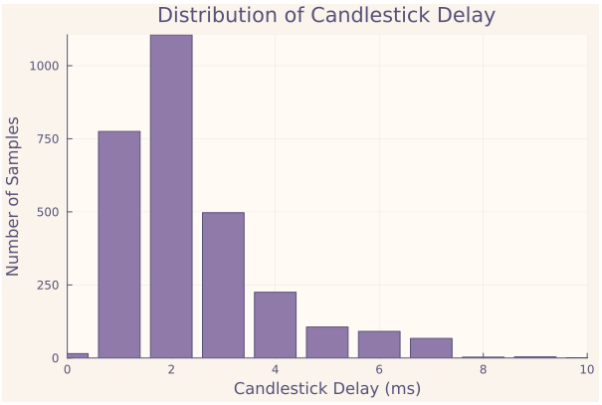
Σχήμα 1: Καθυστέρηση του server: φαίνεται ότι υπάρχει καθυστέρηση η οποία κυμαίνεται κυρίως γύρω από τα 1300msec.



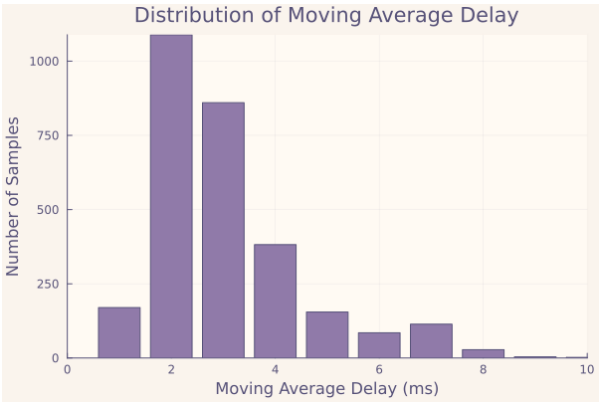
Σχήμα 2: Καθυστέρηση καταγραφής της συναλλαγής: Η καθυστέρηση αυτή φαίνεται να είναι της τάξης των μsec καθώς τις περισσότερες φορές φαίνεται να είναι μηδενική.

Ανεξαρτησία

Το πρόγραμμα έτρεξε στο Raspberry Pi για τουλάχιστον 48 ώρες. Πολλές φορές έτυχε να αποσυνδεθεί ο client σε τυχαίες χρονικές στιγμές από τον server, για άγνωστο λόγο από την μεριά του server. Για τον λόγο αυτό χρησιμοποιήθηκε μια while loop χάρη στην οποία ο client επιχειρούσε κάθε φορά την



Σχήμα 3: Καθυστέρηση υπολογισμού και καταγραφής του candlestick.



Σχήμα 4: Καθυστέρηση υπολογισμού και καταγραφής του moving average. Σαφώς μεγαλύτερη από την καθυστέρηση των candlesticks αφού υπολογίζεται και αποθηκεύεται μετά από αυτά.

επανασύνδεση του. Η προσπάθεια επανασύνδεσης πραγματοποιείται μετά από ένα χρονικό διάστημα 25 δευτερολέπτων προκειμένου να έχει αναγνωριστεί από την μεριά του server η αποσύνδεση του client και η κατάργηση της εγγραφής στα σύμβολα πριν την επανασύνδεση και την επανεγγραφή. Είναι ναι μεν πιθανή η απώλεια δεδομένων σε αυτόν τον χρόνο, όμως κρίθηκε σημαντικότερο να υλοποιηθεί μια σχεδίαση που θα μπορεί με αποτελεσματικότητα να επανασυνδέεται αυτόματα στον server. Επίσης αξίζει να σημειωθεί ότι κατά την αποσύνδεση από τον server μπορεί να μην ληφθούν δεδομένα για παραπάνω από 30 δευτερόλεπτα καθώς υπάρχει πιθανότητα να μην καταφέρει να ξανασυνδεθεί κατευθείαν αλλά να κάνει παραπάνω προσπάθειες επανασύνδεσης. Κατά την διάρκεια της εκτέλεσης του προγράμματος μέσω ssh παρατηρήθηκαν κάποιες από τις αποσυνδέσεις, ωστόσο ο client κατάφερνε με επιτυχία να επανασυνδεθεί στον server. Παρόλο που δεν κατάφερε να συνδεθεί με την πρώτη προσπάθεια στον server, χάρη στην while loop ήταν σε θέση να κάνει πολλαπλές προσπάθειες μέχρι να καταφέρει να επανασυνδεθεί.

Λοιπές Παρατηρήσεις

- Στο τέλος του προγράμματος μετρήθηκε το ποσοστό του χρόνου που η CPU μένει αδρανής το οποίο ήταν 99.16%, ένα επιθυμητό αποτέλεσμα καθώς ιδανικά η CPU θα έπρεπε να μείνει αδρανής για τουλάχιστον το 95% του χρόνου.
- Οι συναλλαγές των μετοχών της Apple και της Amazon γινόταν σε συγκεκριμένες ώρες μέσα στην ημέρα, ενώ οι περισσότερες συναλλαγές που λήφθηκαν αφορούσαν την ισοτιμία των κρυπτονομισμάτων Bitcoin (BTC) και Tether (USDT).
- Σε περίπτωση που μέσα στο λεπτό δεν είχε καταγραφεί κάποια συναλλαγή στα αρχεία των candlesticks καταγραφόταν η αρχικοποιημένη κατάσταση του candlestick, δηλαδή: Starting Price: 0.000000, Ending Price: 0.000000, Highest Price: FLT_MIN, Lowest Price: FLT_MAX, Total Volume: 0.000000.
- Ο κώδικας και τα αρχεία που παράχθηκαν βρίσκονται στο: `https://drive.google.com/drive/folders/1F-MY_8f_EA6MveHUw0ZrIWyHJkn1MdX9?usp=sharing`.