Παράλληλα συστήματα - Χειμερινό ‘22

Εργασία 1.3/10-2-2022

Ομάδα:

sdi2000064 Κωνσταντίνος Κανελλάκης

sdi2000212 Μιχάλης Χατζησπύρου

Υλοποίηση προτεραιότητας σε readers ή writers:

Για τον καθορισμό της προτεραιότητας των readers ή των writers το πρόγραμμα λαμβάνει εκτός από τον αριθμό των νημάτων και έναν χαρακτήρα “r” για να δοθεί προτεραιότητα στους readers και “w” για να δοθεί προτεραιότητα στους writers. Σε σχέση με το αρχικό πρόγραμμα που μας δόθηκε φτιάχτηκε ένα νέο με όνομα «pth_ll_rwl_impl» και φέρει αλλαγές στην συνάρτηση «Thread_work» ενώ επίσης ορίστηκαν ακόμα 2 νέες συναρτήσεις, οι «Thread_Reader» και «Thread_Writer». Η συνάρτηση Thread_Reader αντιστοιχεί στην λειτουργία των readers ενώ η Thread_Writer σε αυτή των writers. Επίσης το rwlock αντικαταστάθηκε με ένα mutex και δύο μεταβλητές συνθήκης.

Αναλυτικότερα για στην **Thread_Reader** ο reader κλειδώνει τον mutex και ελέγχει αν τουλάχιστον ένας writer τροποποιεί την λίστα. Σε περίπτωση που κάτι τέτοιο ισχύει μπαίνει σε κατάσταση «busy waiting» με τη χρήση της μεταβλητής συνθήκης read_c και του mutex. Η αναμονή αυτή λαμβάνει τέλος όταν δεν υπάρχει writer που να εκτελεί στο κρίσιμο τμήμα. Στη συνέχεια ο reader αυξάνει τον μετρητή count_reads ο οποίος υποδεικνύει τον αριθμό των readers που εκτελούν κρίσιμο τμήμα. Στη συνέχεια ξεκλειδώνει τον Mutex, εκτελεί το κρίσιμο τμήμα και αφού ξανά κλειδώσει τον mutex μειώνει τη μεταβλητή count_reads κατά ένα καθώς βγαίνει από το κρίσιμο τμήμα. Εφόσον οι readers μπορούν να προσπελάζει την λίστα ταυτόχρονα, αλλά με την προϋπόθεση ότι κανένας writer δεν τροποποιεί την λίστα την ίδια στιγμή ενεργοποιούμε, αν υπάρχει, τον επόμενο writer.

Όσον αναφορά την συνάρτηση **Thread_Writer** ο writer κλειδώνει τον mutex και ελέγχει αν τουλάχιστον ένας writer ή reader τροποποιεί ή προσπελάζει την λίστα. Αν κάτι τέτοιο ισχύει μπαίνει σε κατάσταση «busy waiting» με τη χρήση της μεταβλητής συνθήκης writer_c και του mutex. Στη συνέχεια ο writer αυξάνει τον μετρητή count_writes ο οποίος υποδεικνύει τον αριθμό των writes που εκτελούν κρίσιμο τμήμα. Στη συνέχεια ξεκλειδώνει τον Mutex, εκτελεί το κρίσιμο τμήμα και αφού ξανά κλειδώσει τον mutex μειώνει τη μεταβλητή count_writes κατά ένα καθώς βγαίνει από το κρίσιμο τμήμα.

- Αν ο χρήστης έχει επιλέξει να δώσει προτεραιότητα στους readers ελέγχουμε αν περιμένουν readers να προσπελάσουν τη λίστα και τους δίνουμε προτεραιότητα ενεργοποιώντας τους όλους αφού μπορούν να βρίσκονται στο κρίσιμο τμήμα της λίστας ταυτόχρονα. Αν ωστόσο, δεν περιμένει κάποιος reader τότε ενεργοποιείται, αν υπάρχει, μόνο ένας writer από αυτούς που περιμένουν.

- Αν ο χρήστης έχει επιλέξει να δώσει προτεραιότητα στους writers ελέγχουμε αν περιμένουν writers να τροποποιήσουν τη λίστα και δίνουμε προτεραιότητα σε έναν από αυτούς ενεργοποιώντας τον. Αν ωστόσο, δεν περιμένει κάποιος writer τότε ενεργοποιούμε όλους τους readers για ταυτόχρονη προσπέλαση της λίστας.

Τέλος, λαμβάνοντας υπόψιν τα παραπάνω γίνεται αντιληπτό ότι δεν χρειάζεται κάποια επιπλέον συνθήκη στην Thread_Reader όσον αναφορά την εξασφάλιση προτεραιότητας είτε των readers είτε των writers, καθώς καλύπτεται από τις συνθήκες της Thread_Writer.

	1 Threads	2 Threads	4 Threads	8 Threads
50% READS,25% INSERTS,25% DELETES				
orig	6.572792	8.501112	18.07742	19.84418
writers	6.566498	7.344579	7.920464	7.981412
readers	6.624563	7.312857	7.786221	7.81782
80% READS,10% INSERTS,10% DELETES				
orig	1.933532	1.631454	1.99861	4.03934
writers	1.865906	2.190555	2.233164	2.735321
readers	1.919523	2.110774	2.236675	2.565927
20% READS,40% INSERTS,40% DELETES				
orig	11.72713	13.42628	31.93829	32.94567
writers	11.69964	13.01334	13.68752	13.8521
readers	11.75311	12.99729	13.53771	13.74319

Σύμφωνα με τον παραπάνω πίνακα, η αρχική υλοποίηση όσο αυξάνονται τα νήματα καταναλώνει σημαντικά περισσότερο χρόνο για την εκτέλεση του προγράμματος, ενώ στην δική μας υλοποίηση η αύξηση είναι αρκετά μικρή.

Όταν το πλήθος των reads είναι 50% και των writes επίσης 50% βλέπουμε ότι δίνοντας προτεραιότητα είτε στους writers είτε στους readers έχουμε σημαντική βελτίωση της απόδοσης.

Όταν το πλήθος των reads είναι 80% και των writes 20% βλέπουμε ότι η απόδοση βελτιώνεται επίσης σημαντικά δίνοντας προτεραιότητα στους readers.

Όταν το πλήθος των reads είναι 20% και των writes 80% βλέπουμε ότι η απόδοση βελτιώνεται επίσης σημαντικά δίνοντας προτεραιότητα στους writers.

Ο κώδικας αναπτύχθηκε και δοκιμάστηκε στο υπολογιστικό σύστημα linux. Τα scripts της bash επίσης έτρεξαν στα μηχανήματα λινουξ της σχολής ωστόσο τα python scripts έτρεξαν σε προσωπικό υπολογιστή με έκδοση 3.6.13 όπου εγκαταστάθηκαν και οι απαιτούμενες βιβλιοθήκες. Να σημειωθεί ότι οι πίνακες και τα δεδομένα που παρουσιάζονται πιο πάνω μπορεί να παρουσιάσουν μερικές μικρές αλλοιώσεις σε σχέση με την θεωρία (αυτά που δηλαδή θα περιμέναμε να βγουν) καθώς στο μηχάνημα που τρέχαμε τα προγράμματα συχνά ήταν και άλλοι συνδεδεμένοι. Συγκεκριμένα χρησιμοποιήθηκε το μηχάνημα με αριθμό 25 του οποίου τα χαρακτηρίστηκα είναι τα εξής:

Operating system: VERSION="18.04.6 LTS (Bionic Beaver)"

Architecture: x86_64

CPU op-mode(s): 32-bit, 64-bit

CPU(s): 4

Thread(s) per core: 1

Core(s) per socket: 4

Socket(s): 1

Model name: Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz

Cache line size: 64

Compiler version: gcc version 7.5.0