Παραλληλη Επεξεργασια

Εργαστηριακη Ασκηση
Εαρινου Εξαμηνου 2010

Στο πλαίσιο του μαθήματος καλείστε να υλοποιήσετε τρεις ασκήσεις, οι οποίες καλύπτουν ένα ευρύ φάσμα μεθόδων παραλληλοποίησης.

Το "Παιχνίδι της Ζωής" (Game of Life)

Το 1970 ο μαθηματικός John Conway επινόησε το "Παιχνίδι της Ζωής" (Game of Life). Το συγκεκριμένο παιχνίδι δεν παίζεται από κάποιον παίκτη, αλλά βάσει των αρχικών συνθηκών εξελίσσεται μόνο του. Επίσης, δεν υπάρχει η έννοια της νίκης ή της ήττας στο παιχνίδι. Η εξέλιξη του παιχνιδιού καθορίζεται από συγκεκριμένους κανόνες που θα παραθέσουμε στην συνέχεια. Το "σύμπαν" του παιχνιδιού είναι ο δισδιάστατος χώρος. Για πρακτικούς λόγους, θα χρησιμοποιήσουμε ένα πεπερασμένο τμήμα του σύμπαντος και θα το αναπαραστήσουμε ως έναν δισδιάστατο πίνακα. Κάθε στοιχείο του πίνακα μπορεί να είναι είτε "ζωντανό" (alive) είτε "νεκρό" (dead). Οι αρχικές συνθήκες ορίζουν ποια στοιχεία θα είναι ζωντανά και ποια νεκρά κατά την εκκίνηση του παιχνιδιού. Σε κάθε βήμα του παιχνιδιού ο σκοπός είναι να καθορίσουμε αν την επόμενη χρονική στιγμή κάθε στοιχείο του πίνακα είναι ζωντανό ή νεκρό. Αυτό καθορίζεται από την τρέχουσα κατάσταση κάθε στοιχείου και την κατάσταση των 8 γειτόνων του. Οι κανόνες που καθορίζουν αν ένα στοιχείο θα είναι ζωντανό ή νεκρό στην επόμενη φάση είναι οι εξής:

i) Αν ένα ζωντανό στοιχείο έχει λιγότερους από 2 ζωντανούς γείτονες, τότε πεθαίνει από μοναξιά.

ii) Αν ένα ζωντανό στοιχείο έχει περισσότερους από 3 ζωντανούς γείτονες, τότε πεθαίνει λόγω υπερπληθυσμού.

iii) Αν ένα ζωντανό στοιχείο έχει 2 ή 3 ζωντανούς γείτονες, τότε παραμένει ζωντανό και στην επόμενη φάση.

iv) Αν ένα νεκρό στοιχείο έχει ακριβώς 3 ζωντανούς γείτονες, τότε ζωντανεύει και το ίδιο.

Θα πρέπει να γίνει κατανοητό πως το πλήθος των ζωντανών και νεκρών γειτόνων υπολογίζεται πάντα πριν εφαρμόσουμε τους παραπάνω κανόνες στους γείτονες. Με λίγα λόγια, πρέπει να βρούμε πρώτα όλα τα στοιχεία που θα αλλάξουν, πριν πάμε και τα αλλάξουμε στον πίνακα. Για να επιτευχθεί αυτό, μπορείτε να ακολουθήσετε μια από τις παρακάτω μεθόδους:

(α') Χρήση ενός δεύτερου πίνακα που θα έχει το ίδιο μέγεθος με τον πρώτο πίνακα που αναπαριστά το σύμπαν. Στον πίνακα αυτό θα τοποθετούμε τα καινούρια ζωντανά και νεκρά στοιχεία, όπως υπολογίζονται βάσει του πρώτου πίνακα. Στην επόμενη φάση, οι πίνακες θα πρέπει να αλλάζουν ρόλους, δηλαδή ο δεύτερος πίνακας θα χρησιμοποιείται ως τρέχων πίνακας και στον πρώτο πίνακα θα πρέπει να τοποθετούμε τις νέες καταστάσεις των στοιχείων.

(i) Η εναλλαγή των πινάκων μπορεί και αυτή να γίνει με δύο τρόπους. Ο πρώτος και απλούστερος στην σύλληψη είναι να αντιγράφουμε κάθε φορά τα στοιχεία του δεύτερου πίνακα στον πρώτο πίνακα και να επαναλαμβάνουμε την προηγούμενη διαδικασία υπολογισμού. Ωστόσο, αν οι πίνακες είναι μεγάλοι τότε απαιτείται αρκετός χρόνος για την αντιγραφή. Αν αποφασίσετε στα πλαίσια της άσκησης να χρησιμοποιήσετε αυτόν τον τρόπο, τότε η αντιγραφή θα πρέπει να γίνεται παράλληλα.

(ii) Ο δεύτερος τρόπος είναι να χρησιμοποιήσετε δείκτες (pointers) για να εναλλάσσετε και να προσπελαύνετε τους πίνακες. Μετά από κάθε φάση, θα απαιτείται μόνο η εναλλαγή των δεικτών για να περάσετε στην επόμενη φάση.

(β΄) Αντιγραφή μόνο των στοιχείων που βρίσκονται στα όρια μεταξύ τμημάτων που επεξεργάζονται διαφορετικά νήματα. Στην περίπτωση αυτή πρέπει σε κάθε νήμα να δημιουργείται κατάλληλους buffers, στους οποίους θα αντιγράφετε τα στοιχεία του πρώτου πίνακα που απαιτούνται για τον υπολογισμό των νέων στοιχείων στα όρια μεταξύ των τμημάτων. Με τον τρόπο αυτό δεν απαιτείται δεύτερος πίνακας και ο υπολογισμός των νέων στοιχείων στο όρια των τμημάτων θα πρέπει να χρησιμοποιεί τις τιμές που έχουν τοποθετηθεί στους buffers. Προφανώς, η λύση αυτή απαιτεί πολύ λιγότερη μνήμη. Ταυτόχρονα όμως απαιτεί και ειδικό χειρισμό στα όρια των τμημάτων και θα πρέπει επίσης να διασφαλίζεται πως κάθε φάση σε ένα νήμα θα ξεκινάει μόνο αφού τα γειτονικά νήματα έχουν αντιγράψει τα στοιχεία που χρειάζονται.

Στα πλαίσια τις άσκησης σας ζητείται να υλοποιήσετε μια παράλληλη έκδοση του παιχνιδιού. Εκτός από τα θέματα που αναφέρθηκαν παραπάνω, θα πρέπει να λάβετε υπ' όψη σας τα εξής:

1. Το παιχνίδι θα πρέπει να τρέχει για 100 φάσεις.
2. Αποφασίστε για το μέγεθος των πινάκων.
3. Αποφασίστε για τις αρχικές καταστάσεις των κελιών. Στο Διαδίκτυο υπάρχουν πολλές ενδιαφέρουσες περιπτώσεις αρχικών καταστάσεων, που οδηγούν σε ενδιαφέρουσες εξελίξεις του παιχνιδιού. Μια πηγή που μπορείτε να χρησιμοποιήσετε για το σκοπό αυτό είναι η `http://www.bitstorm.org/gameoflife/lexicon`
4. Υλοποιήστε το πρόγραμμά σας με τρόπο ώστε να μπορεί να εκτελεστεί σωστά για κάθε αριθμό νημάτων.
5. Αποφασίστε με ποιό τρόπο θα αναθέτετε τμήματα του πίνακα σε νήματα.
6. Να είστε ιδιαίτερα προσεκτικοί στο τέλος κάθε φάσης. Δεν επιτρέπεται νήματα να ξεκινήσουν την επόμενη φάση, πριν όλα τα νήματα ολοκληρώσουν την τρέχουσα φάση. Σκεφτείτε ποιος είναι ο καλύτερος τρόπος συγχρονισμού για να πραγματοποιηθεί αυτό.
7. Μετά την ολοκλήρωση κάθε φάσης να εμφανίζεται στην οθόνη η εξέλιξη του παιχνιδιού (π.χ., με χρήση του χαρακτήρα `x` για τα ζωντανά κελιά και του κενού χαρακτήρα για τα νεκρά κελιά).
8. Για την εκτέλεση ενός παιχνιδιού σε πίνακα NxN θα πρέπει να χρησιμοποιηθεί ένας πίνακας $(N+2)x(N+2)$, του οποίου τα εξωτερικά στοιχεία δεν θα μεταβάλλονται (συνοριακές συνθήκες).

Ο αλγόριθμος του Prim

Ο αλγόριθμος του Prim υπολογίζει ένα Ελάχιστο Γεννητικό Δέντρο (Minimum Spanning Tree - MST) σε έναν μη κατευθυνόμενο γράφο, του οποίου οι ακμές έχουν βάρη. Με λίγα λόγια, ο αλγόριθμος βρίσκει ένα υποσύνολο των ακμών του γράφου που σχηματίζουν ένα δέντρο. Το δέντρο περιλαμβάνει όλες τις κορυφές του γράφου και το άθροισμα των βαρών των ακμών είναι το ελάχιστο δυνατό. Αν συμβολίσουμε το σύνολο των κορυφών με V και το σύνολο των ακμών με E, τότε ο ψευδοκώδικας για τον αλγόριθμο είναι ο παρακάτω:

- Αρχικοποίηση: Θέσε $V_{New} = \{x\}$, όπου x μια τυχαία κορυφή του γράφου και $E_{New} = \{\}$
- Επανάλαβε έως ότου $V_{New} = V$

  i) Επίλεξε μια ακμή (u, v) με το ελάχιστο βάρος, έτσι ώστε $u \in V_{New}$ και $v \notin V_{New}$ (αν υπάρχουν περισσότερες ακμές με το ίδιο ελάχιστο βάρος επίλεξε μια, αλλά πάντα με συνεπή τρόπο).

ii) Πρόσθεσε v στο V_{New} και (u, v) στο E_{New}

Μετά την εκτέλεση του αλγορίθμου τα σύνολα V_{New} και E_{New} αναπαριστούν το Ελάχιστο Γεννητικό Δέντρο.

Καλείστε να υλοποιήσετε μια παράλληλη έκδοση του αλγορίθμου. Η αναπαράσταση του γράφου προτείνετε να γίνει με adjacency matrix (πίνακα γειτνίασης).

Υλοποίηση read/write locks

Κατά την παραλληλοποίηση εφαρμογών απαιτούνται σε σημεία του κώδικα προσπελάσεις σε κοινές μεταβλητές. Παρατηρείται συχνά το φαινόμενο οι μεταβλητές αυτές να πρέπει τις περισσότερες φορές απλά να διαβαστούν και σπάνια να πρέπει να αλλάξει η τιμή τους. Στις περιπτώσεις αυτές η χρήση απλών μεταβλητών κλειδώματος (locks) αποδεικνύεται πως δεν είναι η πλέον αποδοτική μέθοδος συγχρονισμού. Για τον λόγο αυτό έχει εισαχθεί η έννοια των *read/write locks*. Όταν ένα νήμα θέλει να διαβάσει την τιμή μιας μεταβλητής, τότε κλειδώνει την αντίστοιχη μεταβλητή κλειδώματος (lock) ως *reader*. Πολλαπλοί *readers* επιτρέπεται να προσπελαύνουν ταυτόχρονα την κοινή μεταβλητή. Όταν ένα νήμα θέλει να αλλάξει την τιμή της κοινής μεταβλητής, τότε κλειδώνει την αντίστοιχη μεταβλητή κλειδώματος ως *writer*. Αν την χρονική στιγμή εκείνη υπάρχουν *readers*, τότε ο *writer* πρέπει να περιμένει να ξεκλειδώσουν όλοι οι *readers* την μεταβλητή κλειδώματος. Αμέσως μετά ο *writer* επιτρέπεται να συνεχίσει και να αλλάξει την τιμή της μεταβλητής. Αν στο ενδιάμεσο κάποιο νήμα θέλει να γίνει *reader*, τότε πρέπει να περιμένει έως ότου ο *writer* ξεκλειδώσει την μεταβλητή κλειδώματος. Φυσικά, δύο ή περισσότεροι *writers* δεν επιτρέπεται να προσπελαύνουν ταυτόχρονα την κοινή μεταβλητή.

Στα πλαίσια της άσκησης καλείστε να υλοποιήσετε συναρτήσεις που να υλοποιούν ένα read/write lock. Θα πρέπει να χρησιμοποιήσετε τις υπάρχουσες συναρτήσεις του προτύπου POSIX που υλοποιούν μεταβλητές αμοιβαίου αποκλεισμού locks και και μεταβλητές υπό συνθήκη condition variables. Η υλοποίηση σας θα πρέπει να προσφέρει τις παρακάτω συναρτήσεις:

my_rw_init(my_rw_t *lockvar) Αρχικοποιεί την μεταβλητή κλειδώματος lockvar.

my_rw_readlock(my_rw_t *lockvar) Κλειδώνει την μεταβλητή κλειδώματος lockvar ως *reader*.

my_rw_writelock(my_rw_t *lockvar) Κλειδώνει την μεταβλητή κλειδώματος lockvar ως *writer*.

my_rw_readunlock(my_rw_t *lockvar) Ξεκλειδώνει την μεταβλητή κλειδώματος lockvar που είχε κλειδωθεί ως *reader* από το συγκεκριμένο νήμα.

my_rw_writeunlock(my_rw_t *lockvar) Ξεκλειδώνει την μεταβλητή κλειδώματος lockvar που είχε κλειδωθεί ως *writer* από το συγκεκριμένο νήμα.

my_rw_destroy(my_rw_t *lockvar) Καταστρέφει την μεταβλητή κλειδώματος lockvar. Δεν επιτρέπεται η μετέπειτα χρήση της σε διαδικασίες κλειδώματος/ξεκλειδώματος.

Ο τύπος my_rw_t θα πρέπει να οριστεί ως μια δομή η οποία θα περιέχει τα απαραίτητα πεδία για την υλοποίηση των read/write locks. Προσοχή θα πρέπει να δοθεί στο γεγονός πως ένα νήμα που θέλει να κλειδώσει μια μεταβλητή κλειδώματος ως *writer* θα πρέπει να έχει μεγαλύτερη προτεραιότητα από αντίστοιχα νήματα *readers* που θα προσπαθήσουν να κλειδώσουν την μεταβλητή μετά από αυτό. Για τον έλεγχο της ορθότητας της υλοποίησης θα πρέπει επιπλέον να δημιουργήσετε ένα πρόγραμμα, το οποίο θα φτιάχνει 16 νήματα. Δύο από τα νήματα θα λειτουργούν ως *writer* και θα αυξάνουν 10.000 φορές κατά ένα μια κοινή μεταβλητή, της οποίας η αρχική τιμή θα είναι 0. Τα υπόλοιπα νήματα θα είναι *readers*. Όταν θα διαβάζουν την τιμή της μεταβλητής θα την χρησιμοποιούν σε έναν οποιοδήποτε υπολογισμό, π.χ., την αύξηση της τιμής μιας τοπικής τους μεταβλητής κατά την τιμή που διάβασαν. Στο τέλος της εκτέλεσης του προγράμματος βεβαιωθείτε ότι και τα 16 νήματα θα διαβάζουν την σωστή τελική τιμή της κοινής μεταβλητής.

Παραδοτέα

Για την ολοκλήρωση της εργαστηριακής άσκησης καλείστε να παραδώσετε ένα συμπιεσμένο φάκελο σε μορφή .tar.gz με όνομα PROJECT_PARAL_xx.tar.gz (όπου xx ο αριθμός της ομάδας σας), ο οποίος θα περιέχει:

- Όλα τα αρχεία του πηγαίου κώδικα σας (source code).
- Επιπλέον απαραίτητα αρχεία για την μετάφραση ή την τεκμηρίωση του κώδικα σας (π.χ. Makefiles)
- Τεχνική αναφορά σε μορφή .pdf, .ps, ή .doc στην οποία θα παρουσιάζονται οι μετρήσεις και τα διαγράμματα σας και θα τεκμηριώνονται τα συμπεράσματα σας από την πειραματική αξιολόγηση. Επίσης, θα εκτιμηθεί ο εύστοχος σχολιασμός των πιο αξιοσημείωτων από τη σκοπιά της παράλληλης επεξεργασίας σημείων της υλοποίησης σας.

Στο φάκελο δε θα πρέπει να περιέχονται αρχεία ενδιάμεσης αναπαράστασης .o ή εκτελέσιμα αρχεία.
Τα παραπάνω παραδοτέα τα αποστέλλετε στην διεύθυνση ηλεκτρονικού ταχυδρομείου:

pds@hpclab.ceid.upatras.gr

απαραιτήτως με θέμα: PROJECT_PARAL_xx (όπου xx ο αριθμός της ομάδας σας). Έπειτα από επιτυχή παράδοση θα σας αποσταλεί αυτόματο μήνυμα επιβεβαίωσης.

Τυπικά Ζητήματα

- Οι ασκήσεις θα πραγματοποιηθούν σε ομάδες. Κάθε ομάδα θα πρέπει να αποτελείται υποχρεωτικά από 3 ή 4 άτομα. Η δήλωση των ομάδων γίνεται μέσω της σελίδας του μαθήματος από τις 22/03/2010 έως τις 16/04/2010. σελίδας του μαθήματος.
- Οι ημερομηνίες της παράδοσης της άσκησης και της προφορικής εξέτασης θα ανακοινωθούν μετά την ανακοίνωση του προγράμματος κάθε εξεταστικής, τον Ιούνιο και τον Σεπτέμβριο αντίστοιχα.
- Επικοινωνία για απορίες:

  ➥ Forum μαθήματος στο My.Ceid.

  ➥ Υπεύθυνος καθηγητής: Ελευθέριος Πολυχρονόπουλος (edp@hpclab.ceid.upatras.gr)

  ➥ Επικουρικό: Ιωάννης Βενέτης (iev@hpclab.ceid.upatras.gr)
  Κώστας Καραντάσης (kik@hpclab.ceid.upatras.gr)