Σχεδίαση και Ανάλυση Αλγορίθμων

1η εργασία

Στόχος της εργασίας είναι η εξοικείωση με βασικές παράλληλες αλγοριθμικές τεχνικές και την υλοποίησή τους σε συστήματα κατανεμημένης μνήμης. Συγκεκριμένα θα πρέπει να υλοποιήσετε έναν αλγόριθμο εύρεσης βέλτιστης διαδρομής μεταξύ όλων των κόμβων ενός γραφήματος (all pairs shortest paths) και έναν αλγόριθμο ελαχιστοποίησης του κόστους ενός προϊόντος με μικρή διάρκεια ζωής για την παραγωγή, διακίνηση, αποθήκευση και καταστροφή ληγμένων τεμαχίων με χρήση του προγραμματιστικού μοντέλου MPI.

**1ο πρόβλημα (1 μονάδα)**

Στο πρώτο ερώτημα της εργασίας ζητείται να υλοποιήσετε με χρήση του MPI το πρόβλημα της εύρεσης της βέλτιστης διαδρομής μεταξύ όλων των κόμβων ενός γραφήματος (all pairs shortest paths). Συγκεκριμένα, θα πρέπει να τροποποιήσετε τον σειριακό κώδικα που σας δίνεται και να τον παραλληλοποιήσετε με χρήση του προγραμματιστικού μοντέλου MPI.

Ο αλγόριθμος που έχει χρησιμοποιηθεί βασίζεται στην εξής παρατήρηση. Αν συμβολίζει το μήκος της βέλτιστης διαδρομής μεταξύ των κόμβων και που μπορεί να παραχθεί σε το πολύ βήματα, τότε:

Δηλαδή, το βέλτιστο μονοπάτι μεταξύ των κόμβων και που μπορεί να παραχθεί σε το πολύ βήματα αποτελείται από δύο τμήματα μήκους το πολύ βημάτων, ενός μεταξύ των κόμβων και και ενός μεταξύ των κόμβων και . Κατά συνέπεια, θεωρώντας πως το γράφημα έχει κόμβους, ο αλγόριθμος μπορεί να περιγραφεί ως εξής:

Είσοδος:

l: Μητρώο , που περιέχει τα βάρη των συνδέσεων μεταξύ κόμβων

Έξοδος:

l: Ανανεωμένο μητρώο που περιέχει το βέλτιστο κόστος διαδρομής μεταξύ κόμβων

do

Για κάθε κόμβο j

Για κάθε κόμβο i

lij Κόστος τρέχουσας βέλτιστης διαδρομής μεταξύ i και j

Για κάθε πιθανό ενδιάμεσο κόμβο k

lik Κόστος τρέχουσας βέλτιστης διαδρομής μεταξύ i και k

lkj Κόστος τρέχουσας βέλτιστης διαδρομής μεταξύ k και j

Αν lik + lkj < lij

lij lik + lkj // Ενημέρωσε πως βρέθηκε καλύτερη διαδρομή

while υπάρχουν αλλαγές στο μητρώο l

Πριν ξεκινήσετε την υλοποίηση με χρήση του MPI, εκτελέστε τον κώδικα που δίνεται με διάφορα μεγέθη γραφημάτων (π.χ. 200, 300, 400, ..., 1500) και καταγράψτε τους χρόνους εκτέλεσης και τα checksum που παράγει η εφαρμογή. Η εφαρμογή παράγει εσωτερικά ένα γράφημα με το ζητούμενο πλήθος κόμβων, οπότε δεν χρειάζεται κάτι επιπλέον από την μεριά σας. Ειδικά δε το checksum θα σας βοηθήσει να επιβεβαιώσετε ότι η υλοποίηση σας με MPI είναι σωστή (η υλοποίηση με MPI πρέπει να παράγει το ίδιο checksum με την σειριακή εκτέλεση της εφαρμογής για το ίδιο μέγεθος γραφήματος).

Η μεταγλώττιση της εφαρμογής θα γίνεται με την εντολή:

mpicc –o path path.c mt19937p.c

Η εκτέλεση της εφαρμογής θα γίνεται με την εντολή:

mpirun –np <πλήθος διεργασιών> ./path –n <πλήθος κόμβων γραφήματος>

Στην συνέχεια θα πρέπει να μετρήσετε την απόδοση της εφαρμογής σας όταν χρησιμοποιείσετε διαφορετικά πλήθη διεργασιών. Μετρήστε τον χρόνο εκτέλεσης της εφαρμογής με 1, 2, 4, …, p διεργασίες και δημιουργήστε ένα γράφημα της χρονοβελτίωσης (speedup) για κάθε μέγεθος γραφήματος που χρησιμοποιήσατε ως είσοδο και στην σειριακή εκτέλεση της εφαρμογής. Καταγράψτε τις παρατηρήσεις σας.

**2ο πρόβλημα (1 μονάδα)**

Θεωρείστε ένα προϊόν το οποίο έχει διάρκεια ζωής *J* χρονικών μονάδων. Θεωρώντας πως η ζήτηση για το προϊόν για ένα χρονικό διάστημα *T* > *J* μπορεί να προσεγγιστεί μέσω στοχαστικών διαδικασιών, ζητείται να υπολογιστεί το ελάχιστο δυνατό κόστος για την παραγωγή, διακίνηση, αποθήκευση και καταστροφή ληγμένων τεμαχίων του προϊόντος αυτού.

Η σειριακή εφαρμογή που σας δίνεται λύνει ακριβώς αυτό το πρόβλημα, βασιζόμενο στην προσέγγιση που ακολουθείται στο [1] (το πλήρες κείμενο της δημοσίευσης δίνεται μαζί με τον σειριακό κώδικα της εφαρμογής). Στα πλαίσια της εργασίας θα πρέπει να τροποποιήσετε τον σειριακό κώδικα που σας δίνεται και να τον παραλληλοποιήσετε με χρήση του προγραμματιστικού μοντέλου MPI.

Η μεταγλώττιση της εφαρμογής θα γίνεται με την εντολή:

mpicc –o main main.c

Η εκτέλεση της εφαρμογής θα γίνεται με την εντολή:

mpirun –np <πλήθος διεργασιών> ./main <heuristic>

όπου <heuristic> είναι ένας από τους ευρετικούς αλγόριθμους που παρουσιάζονται στο [1]. Στα πλαίσια της εργασίας μπορείτε να χρησιμοποιήστε τις τιμές από 0 έως και 3, ενώ θα πρέπει να προσαρμόσετε την υλοποίηση όλων των ευρετικών αλγορίθμων για την παράλληλη υλοποίηση σας..

Ως βοήθεια στην επίλυση από μέρους σας του ερωτήματος, αναφέρουμε πως θα πρέπει να τροποποιήσετε τουλάχιστον τις συναρτήσεις main(), FirstOrder\_RoundRobin(), SecondOrderByPrOrder(), SecondOrderByPr(), με τις τελευταίες τρεις να υλοποιούν τους ευρετικούς αλγόριθμους που προαναφέρθηκαν με χρήση των κατάλληλων δεδομένων από αντίστοιχα αρχεία εισόδου.

Θα πρέπει να πάρετε αντίστοιχες μετρήσεις για τους χρόνους εκτέλεσης της εφαρμογής όπως και στο πρώτο ερώτημα και να καταγράψετε τις παρατηρήσεις σας.

**Παραδοτέα**

Η εργασία θα πρέπει να παραδοθεί σε CD καθώς και εκτυπωμένη. Πέρα από τον κώδικα των εφαρμογών που θα αναπτύξετε, η εργασία θα πρέπει να περιλαμβάνει και αναλυτική τεκμηρίωση των τεχνικών που ακολουθήσατε στην υλοποίηση, συμπεριλαμβανομένου και ενός ψευδοκώδικα όπου θα συνοψίζεται η βασική δομή των προγραμμάτων σας.

Η εργασία μπορεί να εκπονηθεί από ομάδα **μέχρι 2 ατόμων** και θα πρέπει να παραδοθεί έως **τη Δευτέρα 29/01/2018** στη θυρίδα του διδάσκοντος (5ος όροφος).

**Βιβλιογραφία**

1. Accelerating an algorithm for perishable inventory control on heterogeneous platforms. A. Gutierrez-Alcoba, G. Ortega, E. M.T. Hendrix, I. Garcia, Journal of Parallel and Distributed Computing 104(2017), pp. 12-18.