Παράλληλα και Διανεμημένα Συστήματα

Εργασία 4

Υλοποίηση του Warshall-Floyd algorithm με χρήση Cuda και MPI

Ονοματεπώνυμο: Νικόλαος Λάτμος
ΑΕΜ: 7912
Επικοινωνία: nlatmosa@auth.gr

1 Εισαγωγή

Σε αντίθεση με την τρίτη εργασία, εδώ θα υλοποιηθεί η επίλυση του προβλήματος εύρεσης της ελάχιστης απόστασης μεταξύ δύο κόμβων μέσω του συνδυασμού CUDA και MPI. Για να επιτευχθεί αυτό είναι απαραίτητο να χωρίσουμε τον αρχικό πίνακα που μας δίνεται σε πολλά κομμάτια. Στη συνέχεια, κάθε κομμάτι θα ανατεθεί σε μία διεργασία η οποία θα αναλαμβάνει την εκτέλεση του αλγορίθμου Warshall-Floyd ανεξάρτητα από τα υπόλοιπες διεργασίες. Αυτή η εκτέλεση θα γίνει στη κάρτα γραφικών μέσω των ειδικών βιβλιοθηκών που μας προσφέρει η Nvidia, την επονομαζόμενη πλατφόρμα CUDA.

Πριν προχωρήσουμε, πρέπει να αναφέρω ότι η υλοποίηση του τρίτου πυρήνα σε σχέση με τη τρίτη εργασία έχει τροποποιηθεί. Επίσης, θα ήθελα να τονίσω πως υπήρξε μία αργή εκτέλεση του MPI στο μηχάνημα με 4 πυρήνες (quatro) με αποτέλεσμα να αναγκαστώ να τρέξω το πρόγραμμα στον διάδη όπο όμως η ταχύτητα εκτέλεσης της cuda ήταν πιο αργή σε σχέση με τον quatro. Για τη μέγιστη απόδοση του προγράμματος ιδανικό θα ήταν να τρέξει σε ένα cluster. Τέλος, οφείλω να επισημάνω πως με τη δημιουργία 32 threads και άνω κατά την εκτέλεση στον διάδη παρουσιάζεται σφάλμα κατά της διανομή μνήμης στη κάρτα γραφικών με αποτέλεσμα να μη καταφέρω να μετρήσω τους χρόνους εύρεσης των ελαχίστων αποστάσεων μεταξύ των κόμβων του αρχικού πίνακα.

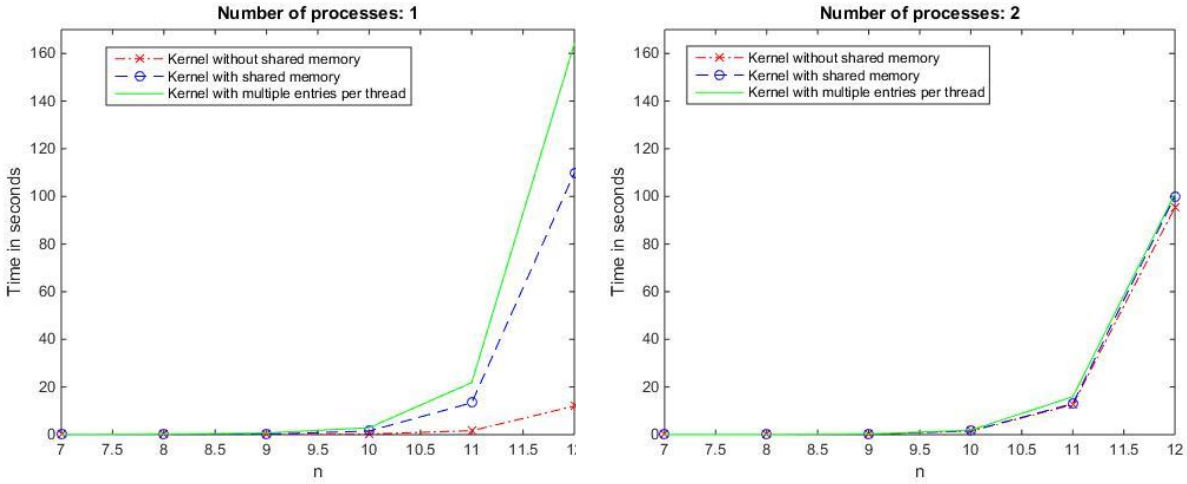
2 Υλοποίηση

Υπενθυμίζουμε ότι η υλοποίηση του Warshall-Floyd αλγορίθμου γίνεται μέσω του ελέγχου της παρακάτω συνθήκης,

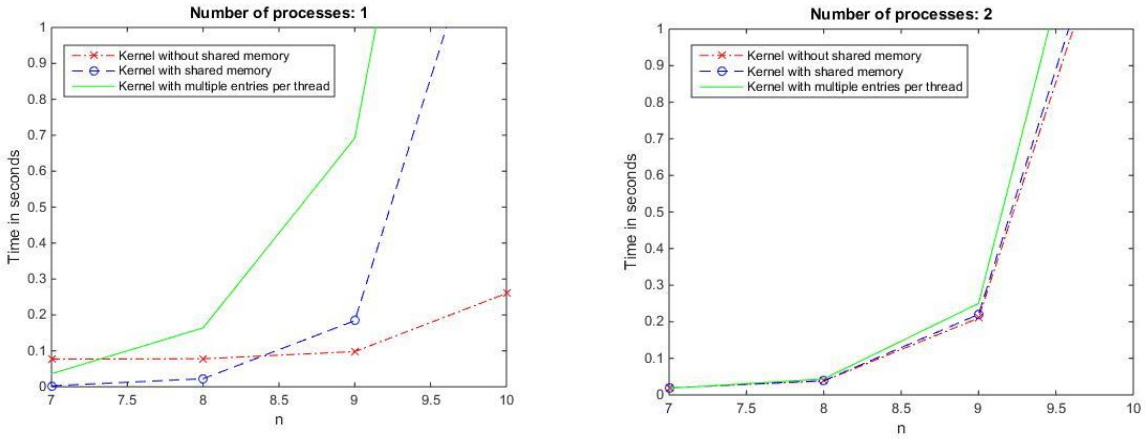
$$a(i,j) = \min[a(i,j), a(i,k) + a(k,j)\ ]$$

Ο αρχικός πίνακας μεγέθους n*n χωρίζεται σε κομμάτια μεγέθους n*n/world_size, όπου world_size είναι ο αριθμός των διεργασιών. Συγκεκριμένα, θεωρήθηκε ως καταλληλότερη επιλογή ο διαχωρισμός του αρχικού πίνακα κατά γραμμές για δύο λόγους. Πρώτον, λόγω της ευκολίας μετατροπής μίας θέσης του δισδιάστατου πίνακα σε μονοδιάστατο και δεύτερον, λόγω του γεγονότος ότι σε κάθε επανάληψη του αλγορίθμου το στοιχείο a(i,k) θα βρίσκεται πάντα στο κομμάτι πίνακα της διεργασίας στην οποία ανήκει. Βέβαια, το στοιχείο α(k,j) συνήθως θα ανήκει σε ξεχωριστή διεργασία. Γίνεται λοιπόν σαφές ότι κάθε διεργασία στην οποία ανήκει η σειρά με συντεταγμένη k, θα πρέπει να μεταδώσει όλα τα στοιχεία αυτής της σειράς στις υπόλοιπες διεργασίες. Η συγκεκριμένη μετάδοση θα λαμβάνει μέρος μία φορά για κάθε βήμα k πριν την τοπική εκτέλεση του αλγορίθμου κάθε διεργασίας ώστε να κατέχει όλα τα στοιχεία που χρειάζεται για τους υπολογισμούς της.

3 Διαγράμματα Χρόνων Εκτέλεσης

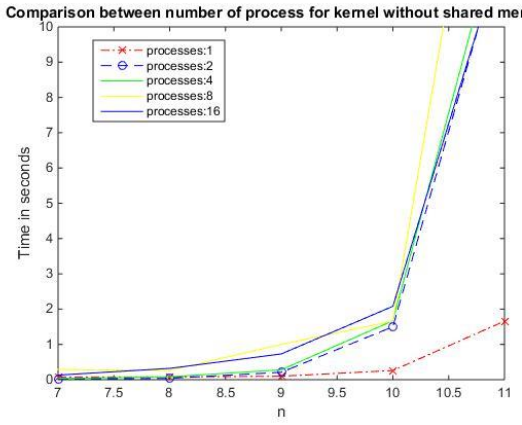


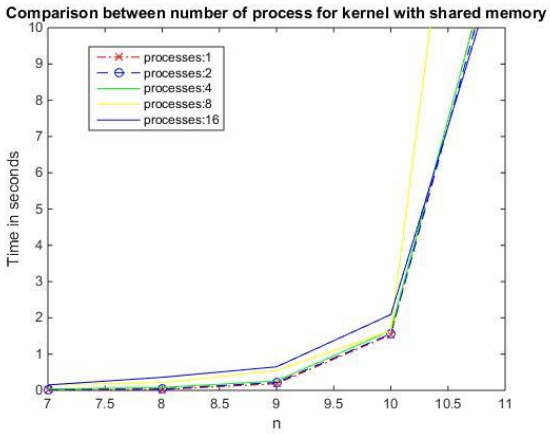
Η πρώτη παρατήρηση που θα ήθελα να κάνω είναι η ραγδαία αύξηση του χρόνου εκτέλεσης στην περίπτωση του πυρήνα δίχως χρήση κοινής μνήμης με δύο διεργασίες σε σύγκριση με την τρίτη εργασία όπου υπάρχει μόλις μία διεργασία. Δύο είναι τα πιθανά προβλήματα, πρώτο η χρήση του MPI και οι χρονικές καθυστερήσεις των επικοινωνιών, ενώ δεύτερο η σειριακή εκτέλεση υπολογισμών από πλευράς της κάρτας γραφικών. Δηλαδή, από τη στιγμή που υπάρχουν 2 διεργασίες θα πρέπει να περιμένει κάποια από τις δύο την ολοκλήρωση των υπολογισμών στη κάρτα γραφικών και εν συνεχεία να προχωρήσει η επόμενη στην εκτέλεση των δικών της υπολογισμών. Ίσως η εκτέλεση του προγράμματος σε έναν cluster να μην εμφάνιζε τόσο μεγάλη απόκλιση. Παρόλα αυτά, βλέπουμε μία βελτίωση τόσο στον δεύτερο πυρήνα όσο και στον τρίτο.



Εδώ βλέπουμε τα ίδια διαγράμματα αλλά για μικρά n. Παρατηρούμε ότι σε αυτή τη περίπτωση μέχρι και για n = 8 η εκτέλεση με δύο διεργασίες είναι ταχύτερη από μία για την περίπτωση του πυρήνα χωρίς κοινή μνήμη.

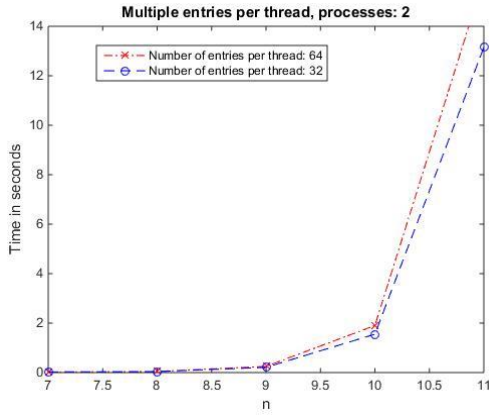
Ας συγκρίνουμε τώρα τους χρόνους εκτέλεσης για διαφορετικό αριθμό διεργασιών.

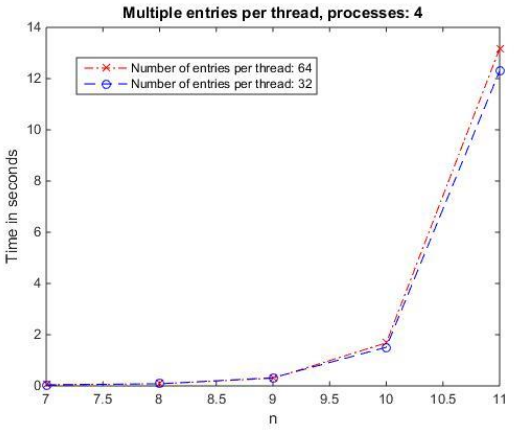


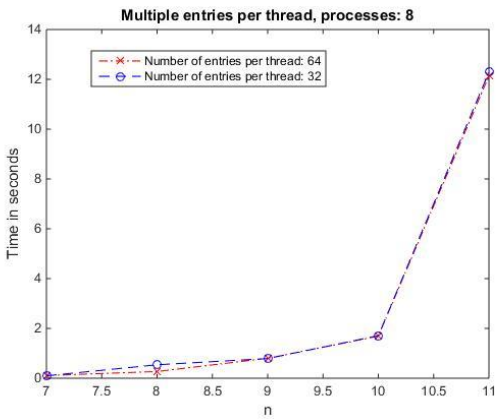


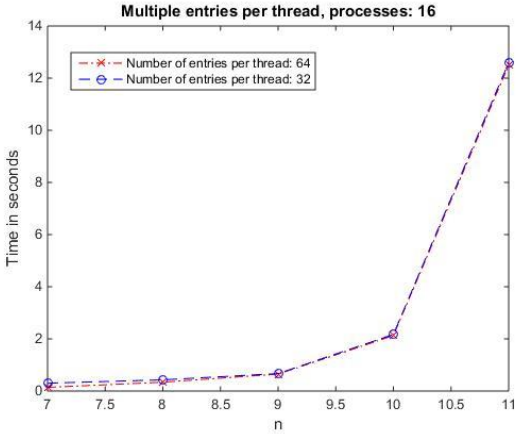
Σε αυτά τα διαγράμματα φαίνονται οι χρόνοι εκτέλεσης για διαφορετικό αριθμό διεργασιών και δύο τύπους πυρήνων. Παρατηρούμε ότι δεν υπάρχει κάποιος ιδανικός αριθμός διεργασιών για την ταχύτερη δυνατή εκτέλεση του αλγορίθμου για όλα τα n.

Τέλος θα παρουσιαστούν τα διαγράμματα για τον τρίτο πυρήνα για διαφορετικό αριθμό κελιών ανά thread.









4 Έλεγχος ορθότητας

Για τον έλεγχο ορθότητας εκτελέστηκε ο σειριακός αλγόριθμος Warshall-Floyd και τα αποτελέσματα αποθηκεύτηκαν σε έναν πίνακα, ο οποίος μετά την εκτέλεση της παράλληλης έκδοσης του αλγορίθμου χρησιμοποιείται για να ελέγξει αν διαφέρει κάποιο στοιχείο.

5 Σχόλια

1. Αδυναμία εκτέλεσης του αλγορίθμου για αριθμό διεργασιών μεγαλύτερο του 16.
2. Το σύστημα πολλές φορές για μεγάλο αριθμό threads και n της τάξης 11,12 «κολλούσε»
3. Η εκτέλεση έγινε στον διάδη λόγω έντονων χρονικών καθυστερήσεων mpi στον quarto
4. Για το compilation του κώδικα πατήστε «make» στον terminal (για Ubuntu)
5. Για την εκτέλεση της εργασίας με μία μόλις διεργασία τρέξτε το αρχείο “project3”
6. Για την εκτέλεση του προγράμματος διαβάστε το αρχείο readme.txt