ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολη ΗΜ&ΜΥ
Παράλληλα Συστήματα Επεξεργασίας
Άσκηση 3 – Τελική Αναφορά
Ακ. Έτος 2013-14

Παπαδημητρίου Κων/νος | ΑΜ: 03108769
Παπαδιάς Σεραφείμ | ΑΜ: 03109193
Ημερομηνία: 27/04/2014

Εισαγωγή

Για την τρίτη άσκηση κληθήκαμε να παραλληλοποιήσουμε τον αλγόριθμο Floyd-Warshall για επεξεργαστές γραφικών. Ο αλγόριθμος FW επιλύει το πρόβλημα εύρεσης ελάχιστων μονοπατιών σε γράφο (APSP). Για γράφο μεγέθους N η χρονική πολυπλοκότητα του αλγορίθμου είναι $O(N^2)$. Αυτό τον καθιστά υπερβολικά αναποδοτικό στη χρήση, για μεγάλα μεγέθη εισόδου. Για τον λόγο αυτό είναι σημαντικό να βρεθούν τεχνικές που να ελαχιστοποιούν τον χρόνο εκτέλεσής του. Παρατηρώντας τον αλγόριθμο FW (naive εκδοχή) βλέπουμε πως τα iterations των δυο εσωτερικών for μπορούν να εκτελεστούν παράλληλα.

Για τον αλγόριθμο FW, εκτός της απλής υπάρχει και η tiled εκδοχή. Η ιδέα στην τελευταία είναι να σπάσουμε τον αρχικό πίνακα γειτνίασης σε ίσους υποπίνακες κατάλληλου μεγέθους (έτσι ώστε αυτοί να χωράνε στην κρυφή μνήμη του επεξεργαστή) και στη συνέχεια να υπολογίσουμε τον FW για κάθε υποπίνακα. Αυτή η εκδοχή διακρίνεται σε τρεις φάσεις:

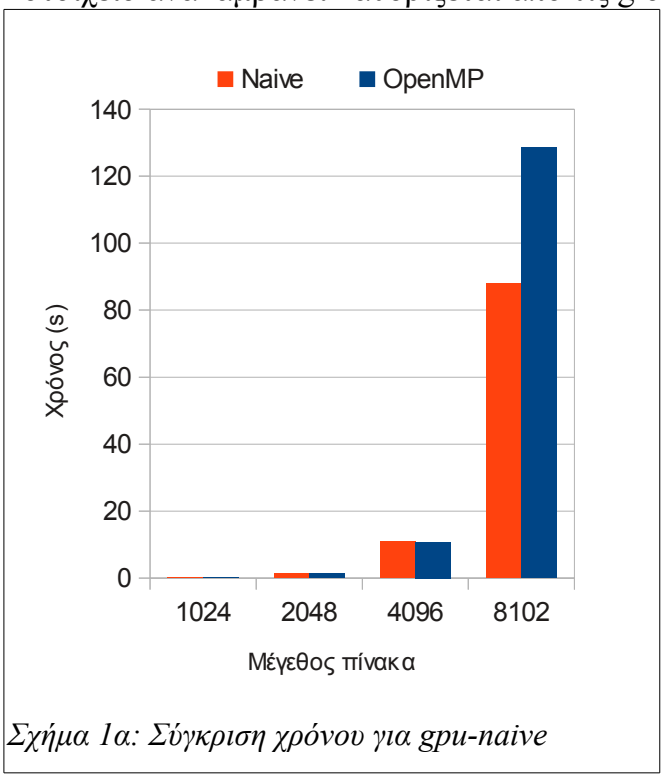
1. Υπολογισμός του tile T_{kk} (όπου *k*, η επανάληψη στην οποία βρισκόμαστε)
2. Υπολογισμός των tiles T_{ik} και T_{ki} (γραμμή και στήλη *k*) με χρήση του T_{kk}
3. Υπολογισμός των υπόλοιπων tile T_{ij} με χρήση των T_{ik} και T_{kj} .

Για την επίλυση της άσκησης χρησιμοποιήσαμε το προγραμματιστικό εργαλείο CUDA C. Στα διαγράμματα γίνονται συγκρίσεις με τα αποτελέσματα που πάρθηκαν από τον παραλληλοποιημένο tiled αλγόριθμο FW με χρήση του εργαλείου OpenMP.

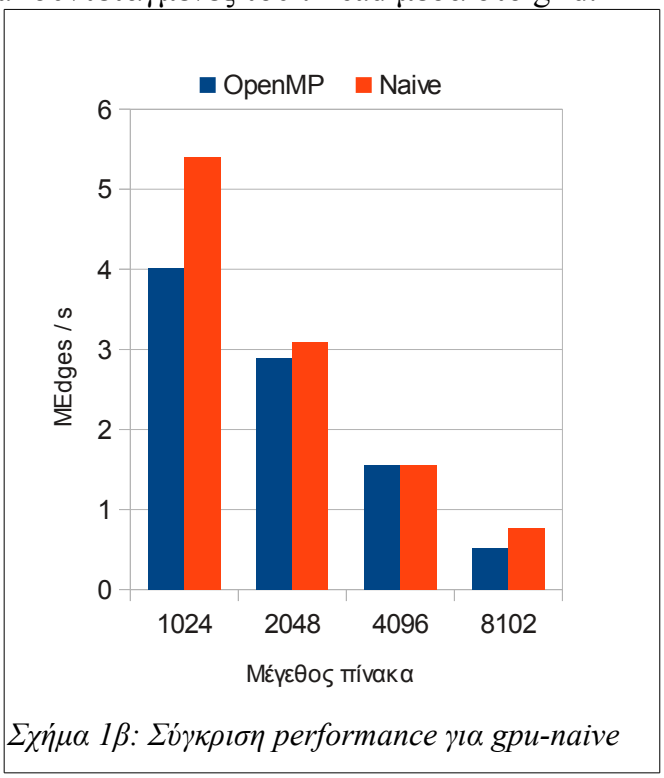
Παραλληλοποίηση

Naive

Η παραλληλοποίηση του απλού αλγορίθμου είναι αρκετά ευθεία και απαιτεί μόνο έναν πυρήνα gpu. Κάθε block του πυρήνα είναι δισδιάστατο και έχει *32* threads σε κάθε διάσταση. Το grid του πυρήνα είναι επίσης δισδιάστατο και έχει *N/32* blocks σε κάθε διάσταση (όπου *N* το μέγεθος του γράφου). Έτσι συνολικά έχουμε N^2 threads, κάθε ένα από τα οποία αναλαμβάνει να υπολογίσει ένα στοιχείο του πίνακα γειτνίασης. Το ποιο στοιχείο αναλαμβάνει καθορίζεται από τις global συντεταγμένες του thread μέσα στο grid.



Σχήμα 1α: Σύγκριση χρόνου για gpu-naive



Σχήμα 1β: Σύγκριση performance για gpu-naive

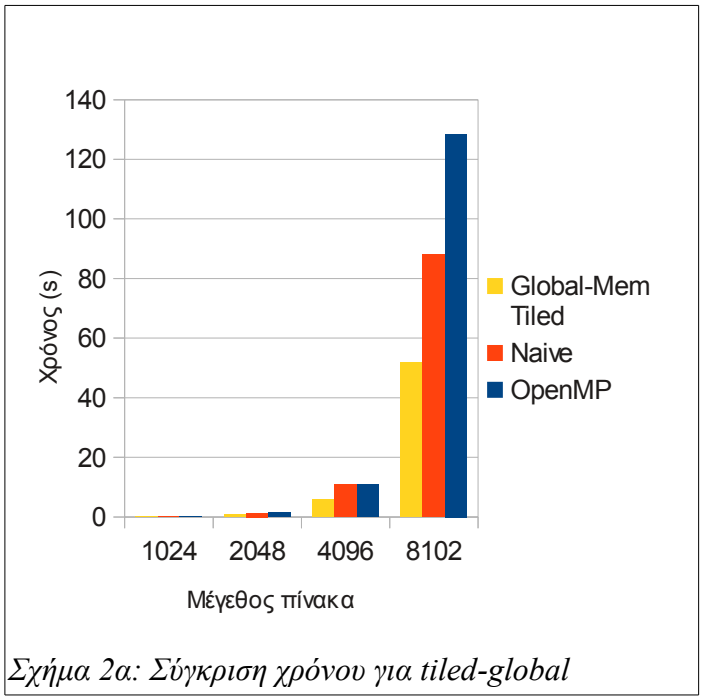
Στα παραπάνω διαγράμματα *(σχήμα 1)* βλέπουμε τη σύγκριση μεταξύ των αποτελεσμάτων από την εκτέλεση του gpu-naive kernel και την εκτέλεση του tiled OpenMP kernel. Παρατηρούμε πως υπάρχει μία σημαντική αλλά όχι εντυπωσιακή βελτίωση. Για να επιτύχουμε καλύτερους χρόνους και υψηλότερη απόδοση πρέπει να υλοποιήσουμε τον tiled αλγόριθμο.

Tiled με χρήση global memory

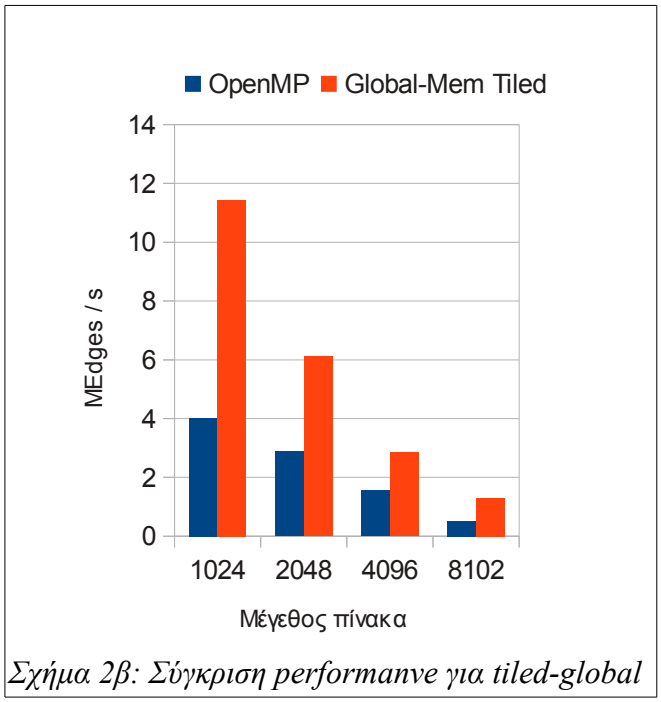
Η παραλληλοποίηση του tiled αλγορίθμου είναι επίσης αρκετά απλή. Κάθε φάση μπορεί να παραλληλοποιηθεί (όπως γίνεται με τον naive αλγόριθμο). Είναι σημαντικό όμως να τηρήται η σειρά μεταξύ των τρειών φάσεων. Για τον λόγο αυτό, τα threads που παραλληλοποιούν την κάθε φάση οφείλουν να συγχρονίζονται. Επειδή μάλιστα το μέγεθος του προβλήματος που υπολογίζει κάθε φάση διαφορετικού μεγέθους, απαιτούνται τρεις διαφορετικοί πυρήνες για την υλοποίηση κάθε φάσης. Το block και για τους τρεις πυρήνες θα είναι δισδιάστατο και θα ισούτε με το μέγεθος του tile. Έτσι τα threads ενός block θα υπολογίζουν ένα tile.

Ο πρώτος πυρήνας υπολογίζει μόνο ένα tile (το T_{kk}) οπότε θα χρησιμοποιεί grid με ένα block. Ο δεύτερος πυρήνας υπολογίζει μία γραμμή και μία στήλη από tiles (T_{ik} και T_{ki}) οπότε θα έχει grid το οποίο θα αποτελείται από δυο γραμμές blocks. Κάθε γραμμή θα έχει *nr* blocks (*nr* ο αριθμός των tile σε μία διάσταση του αρχικού πίνακα γειτνίασης). Τα blocks της πρώτης γραμμής υπολογίζουν τη γραμμή *ik* και τα blocks της δεύτερης γραμμής του grid υπολογίζουν τη στήλη *ki*. Τέλος ο τρίτος πυρήνας θα έχει δισδιάστατο grid το οποίο σε κάθε διάσταση θα έχει nr blocks. Κάθε block αναλαμβάνει να υπολογίσει το αντίστοιχο tile (καθορίζεται από τις συντεταγμένες που έχει το block μέσα στο grid).

Παρακάτω βρίσκονται τα διαγράμματα με τα αποτελέσματα (και τις συγκρίσεις με το OpenMP tiled) που πήραμε από την εκτέλεση του gpu-tiled αλγορίθμου.



Σχήμα 2α: Σύγκριση χρόνου για tiled-global



Σχήμα 2β: Σύγκριση performanve για tiled-global

Παρατηρούμε πως τώρα έχουμε ακόμα μεγαλύτερη απόδοση και καλύτερους χρόνους. Για σύγκριση, στο *σχήμα 2α* βρίσκονται και οι χρόνοι του naive πυρήνα.

Βλέπουμε επίσης την τετραγωνική αύξηση του χρόνου για διαφορετικά μεγέθη εισόδου.

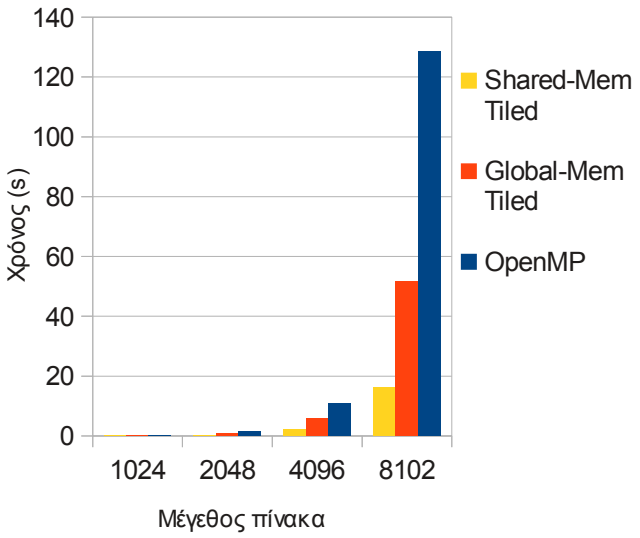
Tiled με χρήση shared memory

Με τον tiled αλγόριθμο καταφέραμε να βελτιώσουμε αρκετά τους χρόνους επίλυσης του προβλήματος. Ωστόσο μπορούμε να κάνουμε χρήση της on-chip shared memory και να πετύχουμε ακόμα μεγαλύτερη απόδοση. Για την υλοποίηση αυτής της έκδοσης του tiled αλγορίθμου ακολουθούμε ακριβώς την ίδια φιλοσοφία που ακολουθήσαμε και στην προηγούμενη έκδοση (τρεις φάσεις που διακρίνονται με καθολικό συγκχρονισμό των νημάτων, ίδια block και grid geometries, κλπ). Ωστόσο, πριν τον υπολογισμό ενός tile το φαίρνουμε (αυτό καθώς και όποια άλλα χρειάζονται) από την κύρια μνήμη στην shared. Αφότου γίνει ο υπολογισμός αυτό αποστέλλεται πίσω στην κύρια μνήμη. Αυτό επιτυγχάνεται με τις συναρτήσεις fetch και send. Αυτές δέχονται στα ορίσματά τους τις συντεταγμένες του tile στον αρχικό πίνακα γειτνίασης. Κάθε thread του block που τις κάλεσε, φέρνει από την (ή στέλνει στην) κύρια μνήμη το αντίστοιχο στοιχείο (καθορίζεται από τις συντεταγμένες του tile και τις συντεταγμένες του thread μέσα στο block). Πριν τον υπολογισμό ενός tile πρέπει να έχουν έρθει όλα τα δεδομένα που χρειάζονται από την κύρια μνήμη. Έτσι τα threads όταν φέρουν τα απαραίτητα tiles από τη μνήμη συγχρονίζονται.

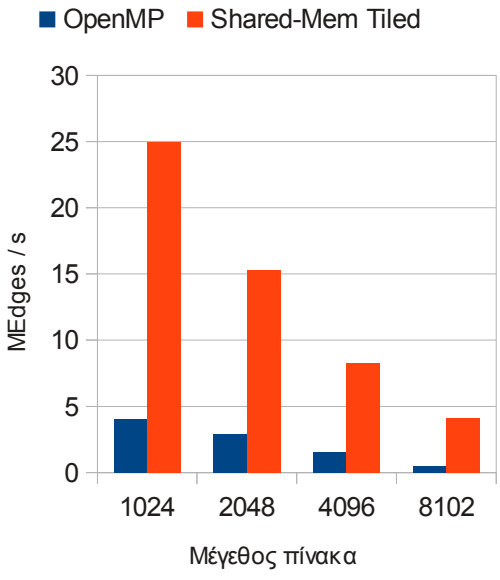
Στον πρώτο πυρήνα δεσμεύεται μόνο ένα tile στη shared memory (σε αυτό θα φέρουμε το T_{kk}). Στον δεύτερο και τον τρίτο δεσμεύονται τρεία tiles (θα έρθουν τα T_{kk} , T_{ik} , T_{ki} και T_{ij} , T_{ik} , T_{kj} αντίστοιχα)

Το ιδανικό μέγεθος tile το βρήκαμε πειραματικά 16 vertices/dim.

Παρακάτω *(Σχήμα 3)* βρίσκονται τα διαγράμματα που συγκρίνουν τον χρόνο και την απόδοση του Tiled αλγορίθμου με χρήση της shared memory με τον χρόνο και την απόδοση του Tiled OpenMP. Βλέπουμε τώρα πολύ μεγάλη βελτίωση στην απόδοση.



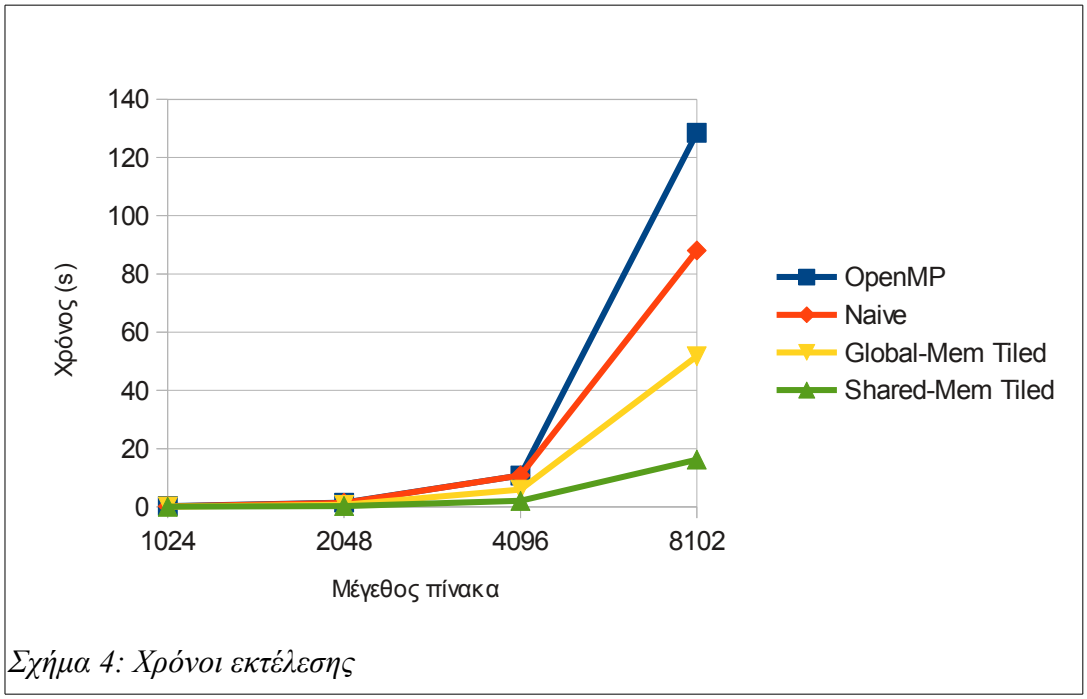
Σχήμα 3α: Σύγκριση χρόνου για tiled-shared



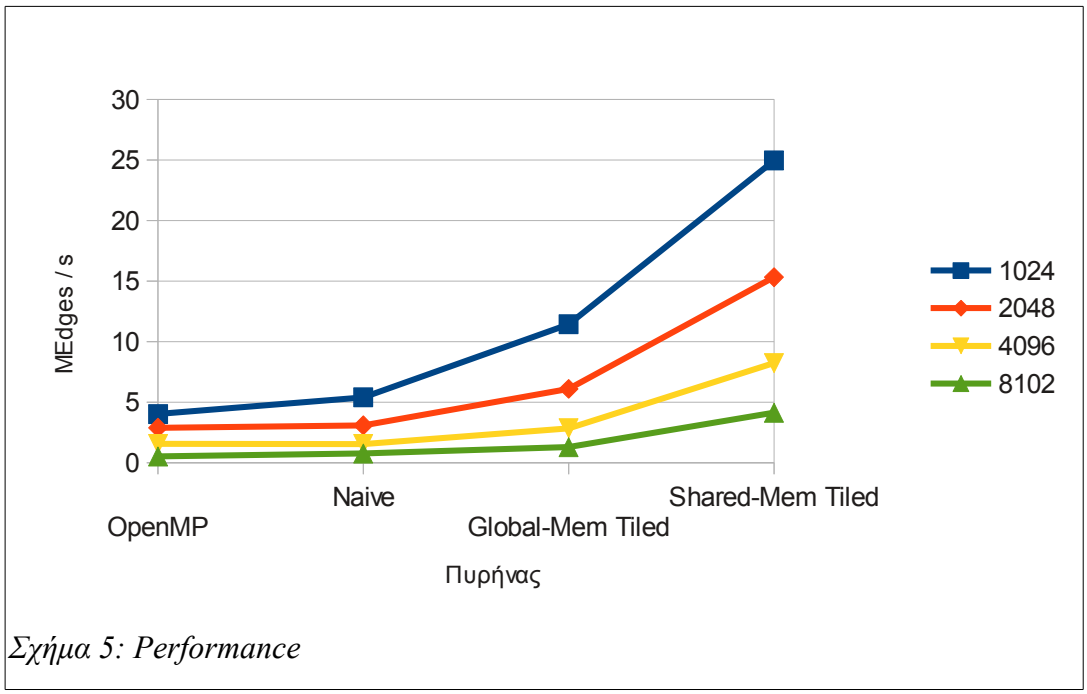
Σχήμα 3β: Σύγκριση performance για tiled-shared

Λοιπά γραφήματα

Παρακάτω παραθέτουμε μερικά γραφήματα που δημιουργήθηκαν με βάση όλες τις μετρήσεις που λάβαμε.



Σχήμα 4: Χρόνοι εκτέλεσης



Σχήμα 5: Performance

Στο *σχήμα 4* βλέπουμε την διακύμανση του χρόνου εκτέλεσης για τα διαφερικά μεγέθη του γράφου. Στο *σχήμα 5* παρατηρούμε τη σημαντική αύξηση της απόδοσης για συγκεκριμένα μεγέθη στους διαφορετικούς πυρήνες.

Συμπεράσματα

Μετά την εκπόνηση της άσκησης, βλέπουμε πώς οι επεξεργαστές γραφικών παρέχουν σημαντικά πλεονεκτήματα στην επίλυση ομάδας προβλημάτων. Είδαμε την απόδοση ακόμα και του απλού αλγορίθμου να είναι σαφώς καλύτερη από την απόδοση του tiled αλγορίθμου υλοποιημένου σε cpu με OpenMP.

Επίσης βλέπουμε πως διαφορετικοί τρόποι υλοποίησης του ίδιου αλγορίθμου μπορούν να φέρουν σημαντικές διαφορές. Χαρακτηριστικό είναι το γεγονός ότι στον tiled αλγόριθμο και για μέγεθος εισόδου 8192 με χρήση μόνο της global memory πετύχαμε χρόνο 51.71s ενώ με χρήση και της shared memory πετύχαμε μόλις 16.22s.

Τέλος, για να γίνει σωστή και βέλτιστη χρήση της gpu απαιτούνται βαθιές γνώσεις της αρχιτεκτονικής, αλλά και καλή εξοικείωση με το προγραμματιστικό περιβάλλον. Ωστόσο, ακόμα και αν είναι αρχάριος κάποιος, μπορεί να επιτύχει ικανοποιητικές αποδόσεις στα προγράμματά του.