Συστήματα Παράλληλης Επεξεργασίας
2013–2014

Άσκηση 3:

Παράλληλος προγραμματισμός σε επεξεργαστές γραφικών

Κόγιας Μάριος-Ευάγγελος (03109064)
Μανούσης Αντώνης (03109031)

Σκοπός

Σκοπός της εργασίας είναι η υλοποίηση του αλγορίθμου Floyd-Warshall, για την εύρεση ελάχιστων μονοπατιών μεταξύ όλων των κόμβων ενός γράφου, σε κάρτες γραφικών και συγκεκριμένα στο περιβάλλον cuda.

Υλοποίηση

Ο συγκεκριμένος αλγόριθμος παραλληλοποιείται εύκολα σε κάρτες γραφικών στην απλή του μορφή, ωστόσο λόγω της έλλειψης χωρικής τοπικότητας τα αποτελέσματα δεν είναι τα βέλτιστα. Για το σκοπό αυτό δημιουργήθηκαν και μετρήθηκαν 3 διαφορετικές υλοποιήσεις του αλγορίθμου. Μια naive υλοποίηση χωρίς κάποια βελτιστοποίηση, μια tiled υλοποίηση και μια shared memory tiled υλοποίηση η οποία κάνει χρήση και της κοινής μνήμης του κάθε SM.
Επιπλέον, δόθηκαν δυο έτοιμες υλοποιήσεις του αλγορίθμου σε OpenMP ώστε να γίνουν οι απαραίτητες συγκρίσεις και έλεγχοι αποτελέσματος.

Naive υλοποίηση

Στην υλοποίηση αυτή χρησιμοποιήθηκε ένας μόνο πυρήνας. Ο πυρήνας αυτός κλήθηκε για κάθε θέση του πίνακα N φορές, όπου N το μέγεθος του πίνακα. Χρησιμοποιήθηκε γραμμικό block και δισδιάστατο grid ώστε να μην υπάρχει πρόβλημα στη δέσμευση πόρων. Με μέγιστο μέγεθος block τα 512 threads μοιράσαμε τον πίνακα σε blocks και σε κάθε block ανατέθηκε ένα κομμάτι του πίνακα. Επιλέξαμε αυτή τη διάταξη και μοίρασμα εργασίας γιατί δεν μας έθετε περιορισμούς σχετικά με το μέγεθος του πίνακα και επιπλέον δεν χρειαζόταν να κάνουμε έξτρα πράξεις μέσα στον πυρήνα για να βρούμε το συγκεκριμένο tid που μας ενδιαφέρει, αφού κάτι τέτοιο θα επιβράδυνε τη διαδικασία. Η επιλογή έγινε αφού πρώτα δοκιμάστηκαν και διαφορετικές διατάξεις με μονοδιάστατα και δισδιάστατα block και grid οι οποίες είχαν χειρότερη επίδοση λόγω των πράξεων που απαιτούνταν για τον υπολογισμό του tid. Τέλος, να σημειωθεί ότι η συγκεκριμένη υλοποίηση δεν χρειάστηκε κάποιο σχήμα συγχρονισμού.
Στο τέλος παραθέτουμε τα διαγράμματα χρόνου και επίδοσης. Παρατηρούμε ότι η naive έκδοση της gpu είναι περίπου 2 φορές καλύτερη σε σχέση με την tiled έκδοση του OpenMP.

Tiled υλοποίηση

Παρόλο που τα αποτελέσματα της naive έκδοσης είναι καλύτερα από τα αντίστοιχα για το OpenMP, ο αλγόριθμος δεν λαμβάνει καθόλου υπόψιν του την τοπικότητα δεδομένων. Για το σκοπό αυτό αναπτύχθηκε μια έκδοση του αλγορίθμου η οποία σπάει τον πίνακα σε μικρότερα κομμάτια τα οποία επεξεργάζεται παράλληλα και τέλος συνθέτει τα κομμάτια αυτά στο τελικό αποτέλεσμα. Στη δικιά μας περίπτωση ο πίνακας χωρίστηκε σε κομμάτια 32x32. Αντίστοιχο ήταν και το μέγεθος του block από threads που χρησιμοποιήθηκε.

Ο αλγόριθμος αυτός περνάει από 3 διαδοχικές φάσεις, κάθε μια από τις οποίες εκτελείται παράλληλα. Για κάθε μια από τις φάσεις αυτές υλοποιήθηκε και ένας διαφορετικός πυρήνας. Κάθε πυρήνας εφαρμόζει και μια διαφορετική συνάρτηση πάνω σε συγκεκριμένο κομμάτι πίνακα. Η συνάρτηση αυτή υπολογίζει ελάχιστα μονοπάτια. Ο ψευδοκώδικας της υλοποίησης δινόταν.
Ο πρώτος πυρήνας κάνει υπολογισμούς στα κομμάτια του πίνακα της διαγωνίου γι'αυτό και χρησιμοποιήθηκε grid με ένα μόνο block με μέγεθος όσο και το κομμάτι του πίνακα.
Ο δεύτερος πυρήνας κάνει υπολογισμούς σε μια συγκεκριμένη γραμμή και μια συγκεκριμένη στήλη του πίνακα. Για το σκοπό αυτό χρησιμοποιήθηκε ένα grid με 2 γραμμές από τις οποίες η μια ανέλαβε τον υπολογισμό της γραμμής και η άλλη τον υπολογισμό της στήλης.
Ο τρίτος πυρήνας κάνει υπολογισμούς σε όλο τον πίνακα με βάση τα προηγούμενα αποτελέσματα. Για το σκοπό αυτό χρησιμοποιήθηκε ένα τετραγωνικό grid.
Οι πυρήνες αυτοί εκτελούνται διαδοχικά τόσες φορές όσα και τα tiles μια γραμμής ή μιας στήλης του πίνακα. Επειδή μέσα στους πυρήνες έχουμε δομές επανάληψης πρέπει να υλοποιηθεί κάποιος συγχρονισμός. Συγκεκριμένα στους 2 πρώτους πυρήνες, έχουμε ένα σημείο συγχρονισμού στο τέλος κάθε επανάληψης. Αντίθετα, στον 3 πυρήνα στον οποίο γίνεται και το μεγαλύτερο μέρος των υπολογισμών δεν υπάρχει αυτή ανάγκη αφού το κάθε block λαμβάνει υπόψιν στους υπολογισμούς μόνο αποτελέσματα των προηγούμενων 2 φάσεων και όχι της ίδιας.
Από τα αποτελέσματα παρατηρούμε ότι η έκδοση αυτή είναι 3-4 φορές καλύτερη από την αντίστοιχη tiled έκδοση του OpenMP.

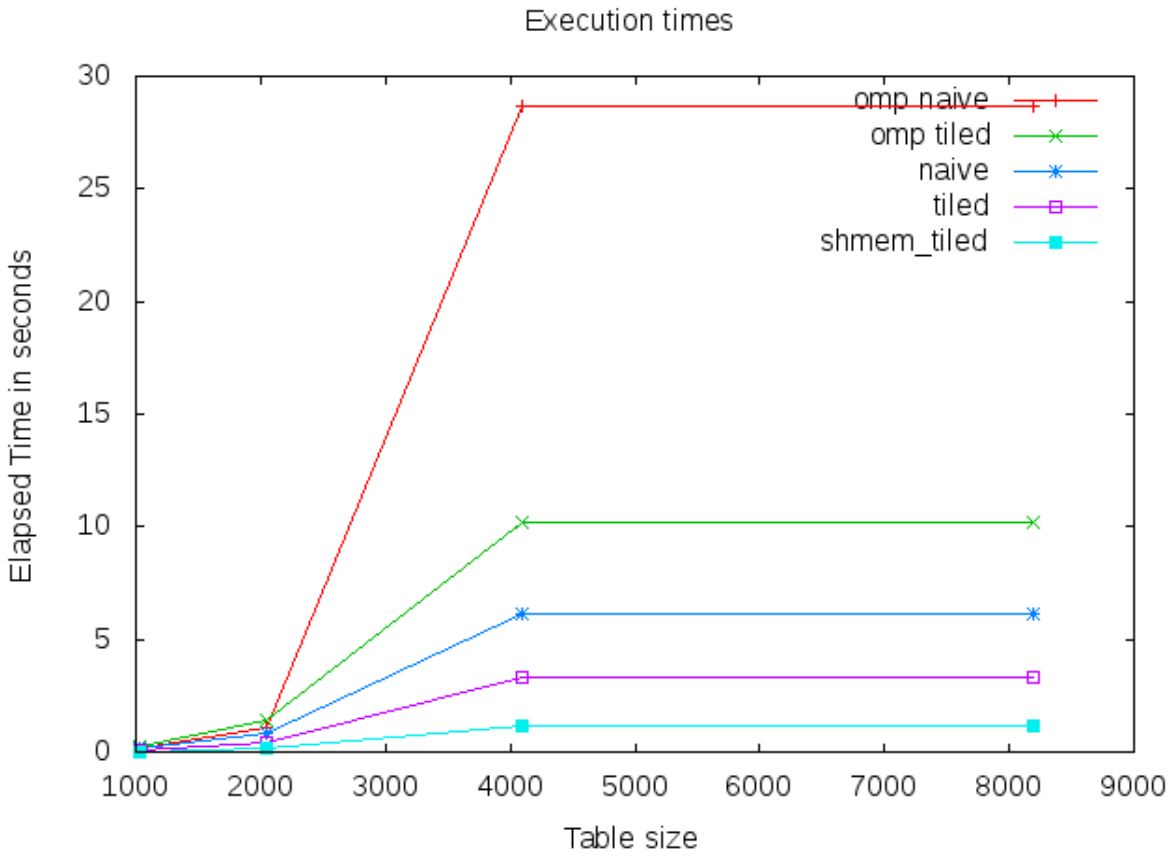
Shared-memory tiled υλοποίηση

Ο συγκεκριμένος αλγόριθμος ακολουθεί την ίδια λογική σε σχέση με τον tiled, δηλαδή λειτουργεί σε τρεις διακριτές φάσεις που υλοποιούνται παράλληλα. Πάλι χρησιμοποιούνται τρεις πυρήνες, κάθε ένας από τους οποίους εφαρμόζει διαφορετική συνάρτηση στον πίνακα με ίδια μεγέθη grid όπως και παραπάνω.
Η βασική διαφορά είναι πως σε κάθε πυρήνα δημιουργούμε από έναν μέχρι τρεις πίνακες τύπου shared με σκοπό να μειώσουμε τις προσβάσεις στη global memory φέρνοντας τα δεδομένα που χρειαζόμαστε για κάθε πυρήνα σε shared on-chip memory. Ο tiled αλγόριθμος όπως αναφέρθηκε και παραπάνω αξιοποιεί την τοπικότητα των δεδομένων και γι αυτό η αποθήκευση δεδομένων σε μια κοινή on-chip δομή που θα χρησιμοποιείται από όλα τα thread ενός block περιμέναμε να δώσει σημαντική βελτίωση στο performance. Για το λόγο αυτό, μετά την αρχικοποίηση κάθε shared πίνακα μεταφέρουμε τα δεδομένα που χρειαζόμαστε μια φορά από τη global memory, στη συνέχεια καλούμε το κύριο σώμα του πυρήνα, το οποίο δε διαφοροποιείται πρακτικά από αυτό της απλής tiled έκδοσης. Τέλος αφού αυτό ολοκληρωθεί μεταφέρουμε τα δεδομένα πίσω στη global memory. Σημαντικός παράγοντας που έπρεπε να ληφθεί υπόψιν είναι αυτός του thread synchronization καθώς έπρεπε να ήμαστε ιδιαίτερα προσεκτικοί ώστε να αποφύγουμε το ενδεχόμενο όπου κάποιο από τα threads προσπαθούσε να διαβάσει/γράψει πληροφορία που δεν ήταν ακόμη διαθέσιμη σε αυτό. Αυτό συμβαίνει καθώς ενώ φαινομενικά τα threads τρέχουν παράλληλα, πρακτικά δεν είναι δυνατόν όλα τα threads να είναι ταυτόχρονα στο ίδιο σημείο της εκτέλεσης του πυρήνα. Για το λόγο αυτό χρησιμοποιήσαμε τη ρουτίνα __syncthreads() της cuda μετά από κομβικά σημεία του πυρήνα, όπως τη μεταφορά των δεδομένων από την global στη shared memory ή μετά την ολοκλήρωση του κυρίου σώματος του πυρήνα ώστε στη συνέχεια να ακολουθήσει μεταφορά πίσω στη global memory. Όπως φαίνεται και από τα αποτελέσματα η βελτίωση που επιτεύχθηκε ήταν ξεκάθαρη με διπλασιασμό του performance σε σχέση με την tiled έκδοση στη GPU.
Σημαντικός παράγοντας για τη συγκεκριμένη υλοποίηση ήταν και η επιλογή του κατάλληλου μεγέθους tile που θα κόψουμε τον πίνακα. Η επιλογή πρέπει να γίνει γνωρίζοντας ότι βάσει της υλοποίησης το μέγεθος του tile είναι ίδιο με το μέγεθος του thread block και κάθε block θα πρέπει να έχει στη διάθεσή του 3 πίνακες tile_size x tile_size στην shared-memory. Επομένως, πρέπει να μεγιστοποιηθεί το μέγεθος του block ώστε στην shared-memory να χωράνε ακριβώς 3 πίνακες. Το συγκεκριμένο μηχάνημα που έγιναν τα πειράματα έχει shared-memory 48 kb. Επομένως, με μέγεθος float 4 byte για να χωρέσουμε 3 πίνακες στη μνήμη το block θα πρέπει να έχει μέγεθος

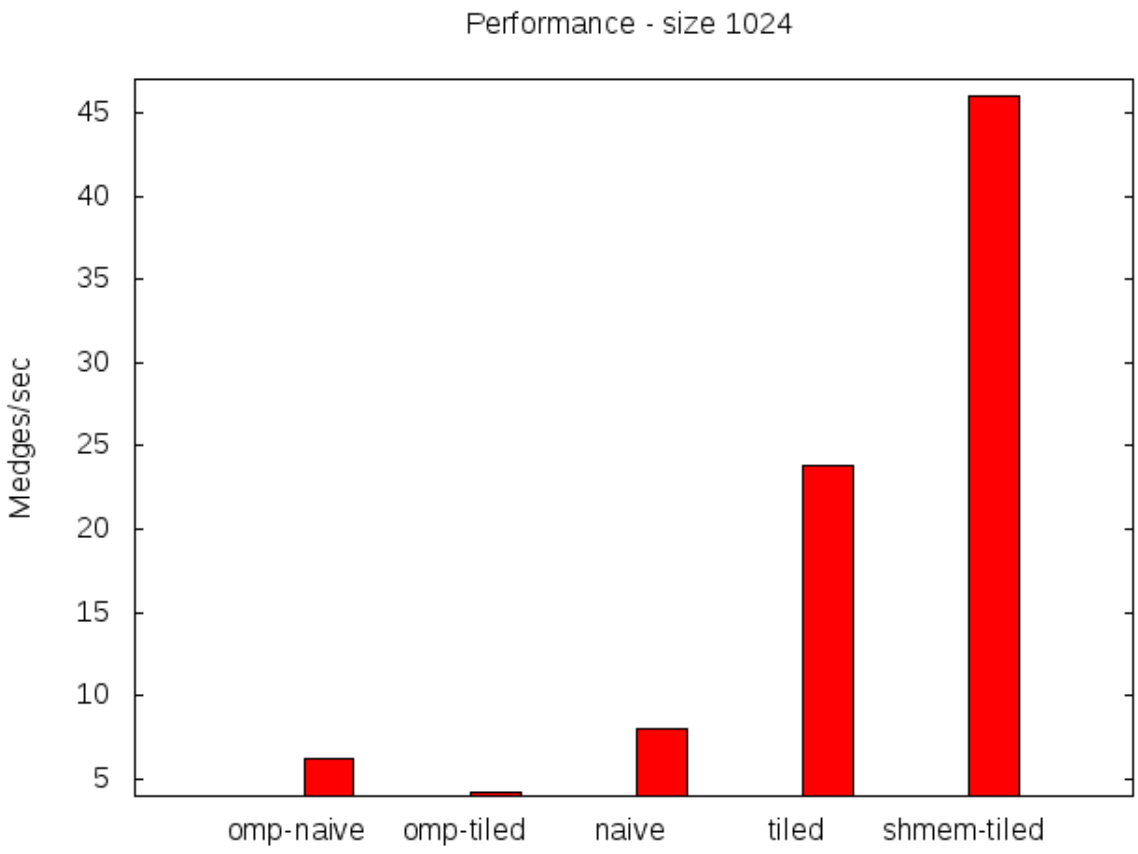
128x128 όμως αυτό δεν είναι εφικτό αφού μπούμε να πάρουμε μόνο 1024 νήματα ανά block.

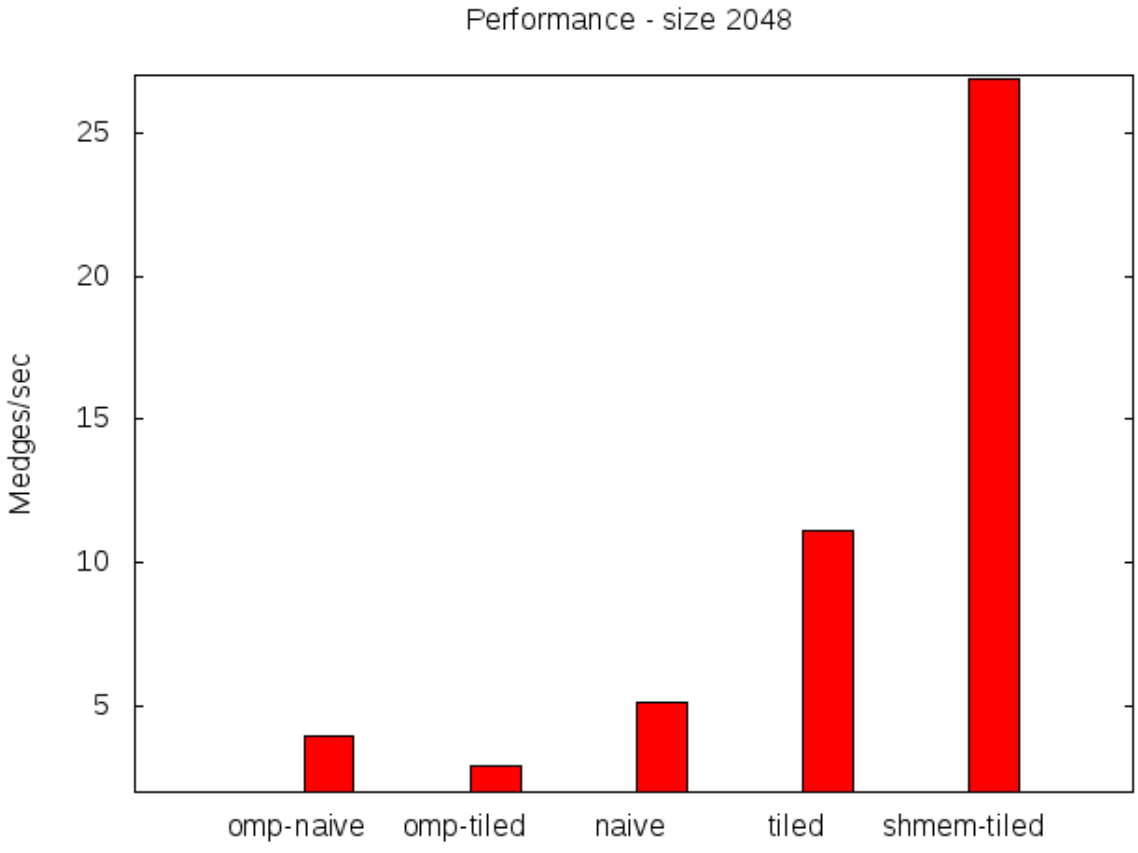
Αποτελέσματα

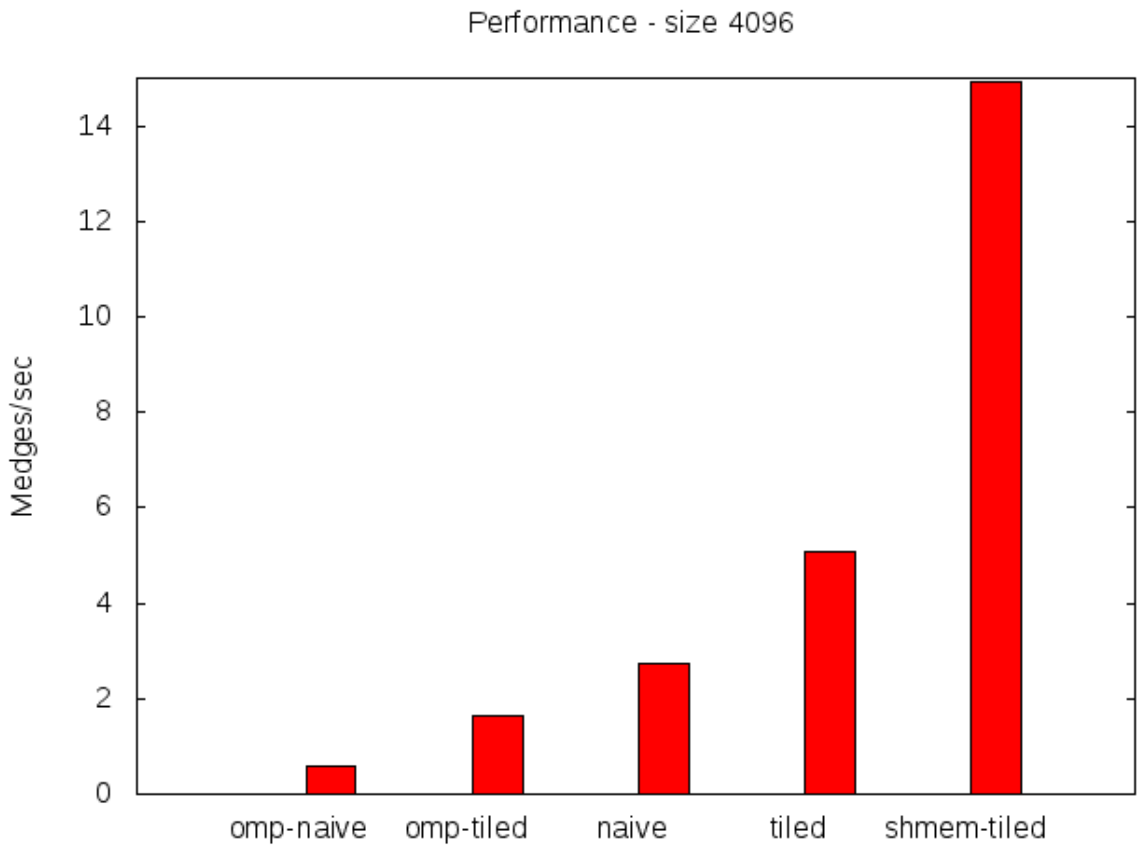
Αρχικά παραθέτουμε ένα διάγραμμα με τους συνολικούς χρόνους εκτέλεσης για διαφορετικά μεγέθη πίνακα.

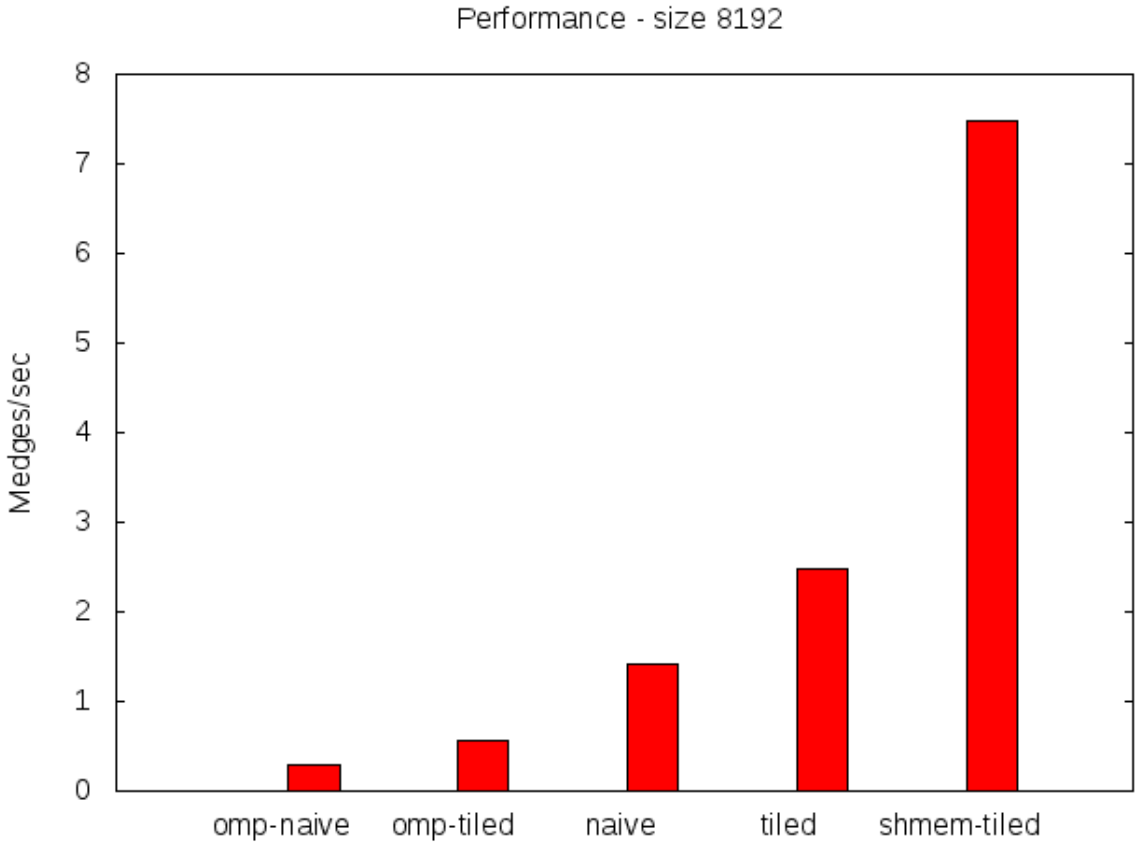


Στη συνέχεια για κάθε μέγεθος πίνακα παραθέτουμε ένα διάγραμμα με το performance metric million edges / second. Αυτό που παρατηρούμε είναι και το αναμενόμενο, οτι δηλαδή η tiled υλοποίηση στη GPU αυξάνει σημαντικά το performance σε σχέση με τη naïve έκδοση, παρόλ'αυτά η ανάγκη για μεταφορά όλων των δεδομένων από τη global memory αποτελεί bottleneck στο performance και στο χρόνο εκτέλεσης. Αυτό φαίνεται να βελτιώνεται ξεκάθαρα με τη χρήση shared δομών με πρακτικά διπλασιασμό του performance. Σημαντική λεπτομέρεια που οδήγησε στην επίτευξη αυτών των αποτελεσμάτων είναι ο τρόπος με τον οποίον τα threads εντός ενός warp αποκτούσαν πρόσβαση στα δεδομένα του block, δηλαδή η διευθυνσιοδότητηση έπρεπε να γίνει με τέτοιο τρόπο ώστε γειτονικά threads να επεξεργαστούν και γειτονικά δεδομένα στη μνήμη και οχι δεδομένα που απείχαν (block_dim)*sizeof(float).


Performance - size 1024
Medges/sec
45
40
35
30
25
20
15
10
5
omp-naive
omp-tiled
naive
tiled
shmem-tiled


Performance - size 2048
Medges/sec
25
20
15
10
5
omp-naive
omp-tiled
naive
tiled
shmem-tiled


Performance - size 4096
Medges/sec
0
2
4
6
8
10
12
14
omp-naive
omp-tiled
naive
tiled
shmem-tiled


Performance - size 8192
Medges/sec
0
1
2
3
4
5
6
7
8
omp-naive
omp-tiled
naive
tiled
shmem-tiled