Παράλληλα και Διανεμημένα Συστήματα

Εργασία 3η

GPU Computing using CUDA

Βλαδίμηρος Στερζεντσένκο

AEM:7777

Email:vlad.sterzentsenko@gmail.com

Υλοποίηση του Game of Life

Οι κανόνες του παιχνιδιού εκφράζονται με την παρακάτω εντολή γραμμένη στην C:

Next_gen=(counter==3)||((counter==2)&&cell_condition);

Όπου counter είναι ο αριθμός των ζωντανών γειτόνων ενός κελιού.Αν ο αριθμός αυτός είναι 3 τότε το κελί θα είναι ζωντανό στην επόμενη γενιά.Αν είναι 2 τότε θα παραμείνει στην προηγούμενη κατάσταση.Σε κάθε άλλη περίπτωση θα είναι νεκρό στην επόμενη γενιά.

Η σειριακή υλοποιήση του Game of Life είναι προφανής και γι'αυτό παραλείπεται.

Πυρήνας 1ος

Υλοποιήση με ένα κελί ανά νήμα της GPU.Κάθε νήμα διαβάζει την κατάστασή του και την κατάσταση των γειτόνων του.Αφού υπολογίσει τον αριθμό των γειτόνων του,γράφει την επόμενη κατάστασή του σε έναν άλλον πίνακα που δίνεται σαν είσοδος στον πυρήνα την επόμενη γενιά.

Συνολικά γίνονται 10 reads (9 γείτονες και η κατάσταση του ίδιου του κελιού) και 1 write στην global memory της GPU κάτι που καθιστά την υλοποιήση αυτή την κατά το δυνατόν πιο αργή,για τα δεδομένα του παράλληλου υπολογισμού σε GPU.

Επιλέχθησαν 512 νήματα ανά μπλόκ,ώστε να επιτεχθεί η μέγιστη αξιοποιήση σε κάθε μπλοκ (πολλαπλάσιο των 32 warps).

Πυρήνας 2ος

Υλοποιήση με πολλαπλά κελιά ανά νήμα της GPU.Κάθε νήμα αναλαμβάνει συνεχόμενα (κατά στήλη) κελιά,έτσι ώστε να υπάρχουν κοινά κελιά σε διαδοχικές επαναλήψεις.Σε 2 διαδοχικά κελιά υπάρχουν 5 κοινοί γείτονες,κάτι που μας γλυτώνει read από την global memory.

Το πρώτο κελί κάνει 10 read και 1 write στην global memory,όπως και στον πυρήνα 1,ωστόσο τα επόμενα κελιά κάνουν 5 read(3 νέοι γείτονες στα δεξιά του κελιού,η κατάσταση του ίδιου και η κατάσταση του προηγούμενου κελιού) και 1 write στην global memory.Αυτό καθιστά τον πυρήνα αυτό αρκετά πιο γρήγορο από τον πρώτο πυρήνα.

Και εδώ χρησιμποιούμε 512 νήματα ανα μπλόκ για τον παραπάνω λόγο.

Πυρήνας 3ος

Υλοποιήση με πολλαπλά κελιά ανά νήμα της GPU με χρήση της shared memory.Χρησιμποιούμε την on-chip shared memory της συσκευής για να δεσμεύσουμε έναν πίνακα 3*threads_per_block.Τα νήματα αναλαμβάνουν συνεχόμενα κελιά κατά γραμμή.

Στάδιο πρώτο:Κάθε νήμα αναλαμβάνει να γράψει σε κάθε στήλη του shared πίνακα.Η πρώτη γραμμή αντιπροσωπεύει τον «πάνω» γείτονα,η δεύτερη την κατάσταση των ίδιων των κελιών και η τρίτη τον «κάτω» γείτονα για κάθε κελί.Στο στάδιο αυτό γίνονται 3 global memory read και 3 shared memory write.

Στάδιο δεύτερο:Αφού γεμίσει με αυτόν τον τρόπο ο πίνακας,τα νήματα είναι σε θέση να υπολογίσουν των αριθμό των ζωντανών γειτόνων.Κατά συνέπεια γίνονται 6 shared memory read και ενα global memory write.

Στάδιο τρίτο:Το νήμα αναλαμβάνει το ακριβώς από κάτω ,στον αρχικό πίνακα, κελί,όπου θα γίνει 1 global memory read και 3 shared memory write για να γεμίσει ο shared πίνακας όπως στο πρώτο στάδιο.Αυτό γίνεται γιατί το ακριβώς από κάτω κελί έχει 2 από τα 3 στοιχεία ίδια στην ίδια στήλη.

Έπειτα υπολογίζεται η κατάσταση του κελιού στην επόμενη γενιά όπως παραπάνω.

Ειδική περίπτωση αποτελούν τα νήματα που αναλαμβάνουν τα κελία που βρίσκονται στην αρχή και στο τέλος του shared πίνακα.Αυτά τα κελιά έχουν γείτονες που δεν θα χρησιμοποιήσει άλλο κελί εντός του μπλόκ και κατά συνέπεια δεν εγγράφονται στον shared πίνακα.Κατά συνέπεια το πρώτο κελί κάνει 3 global read και κάθε επόμενο ακριανό κελί θα κάνει 1 global read σύμφωνα με την παραπάνω λογική.

Προτού μεταβούμε από τον ένα βήμα στο άλλο πρέπει να συγχρονίσουμε τα νήματα,άρα ο χρόνος εκτέλεσης της ρουτίνας είναι ο χρόνος εκτέλεσης του πιο αργού νήματος.

Κατά συνέπεια,στην χειρότερη περίπτωση, το πρώτο κελί ανά νήμα θα κάνει 6 global read,3 shared write,3 shared read και ένα global write.Για τα επόμενα κελιά, θα γίνονται 2 global read,3 shared read,3 shared write και ένα global write για το αποτέλεσμα.

Αναμένουμε αυτός ο πυρήνας να είναι ο πιο γρήγορος.

Εδώ χρησιμοποιούνται 500 νήματα ανά μπλόκ,ώστε κάθε γραμμή να μπορεί να χωριστεί χωρίς υπόλοιπα.

Επιλογή βέλτιστων παραμέτρων.

Για τους πυρήνες 2 και 3,η επιλογή την παραμέτρου που αντιπροσωπεύει το πόσα κελιά θα αναλάβει κάθε νήμα εγίνε με δοκιμές.Για τον $2^{ο}$ πυρήνα η βέλτιστη απόδοση παρουσιάζεται με 4 κελιά ανά νήμα ενώ για τον $3^{ο}$ πυρήνα με 8 κελιά ανά νήμα.Η παρουσίαση των αποτελεσμάτων θα γίνει με τις συγκεκριμένες τιμές.

Για τιμές μεγαλύτερες από τις παραπάνω η απόδοση πέφτει.Αυτό γίνεται γιατί τα νήματα της GPU προορίζονται να είναι βραχύβια σε αντίθεση με αυτά της CPU.

Παρουσίαση Αποτελεσμάτων

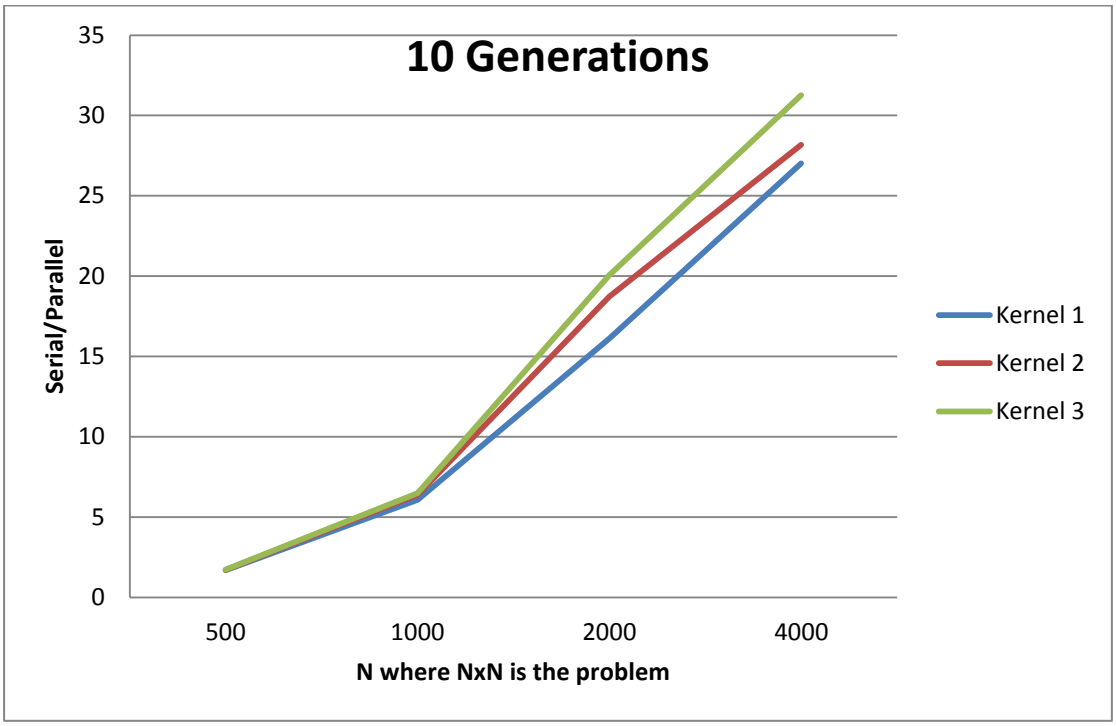
Θα παρουσιαστεί πίνακας με τους χρόνους εκτέλεσης των πυρήνων καθώς και του σειριακού κώδικα.Επίσης θα παρουσιαστούν γραφικές παραστάσεις με κατακόρυφο άξονα τον χρόνο σειριακής εκτέλεσης/χρόνο εκτέλεσης πυρήνα.

Χρόνοι εκτέλεσης (seconds)

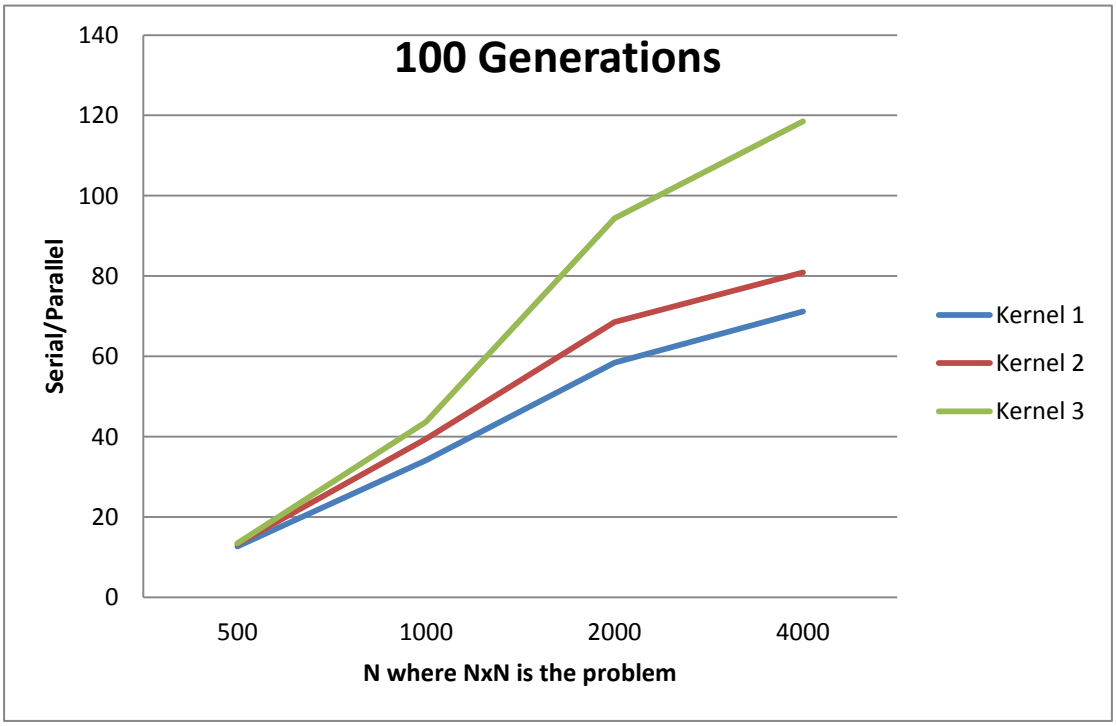
Code \generations	10	100	1000
	N=500		
Serial Code	0.086730	0.765047	7.565211
First Kernel	0.051120	0.060201	0.147207
Second Kernel	0.050341	0.057588	0.124454
Third Kernel	0.050235	0.056897	0.118420
	N=1000		
Serial Code	0.355817	3.135175	30.985524
First Kernel	0.058587	0.091807	0.407027
Second Kernel	0.055531	0.079442	0.311158
Third Kernel	0.054712	0.071715	0.239414
	N=2000		
Serial Code	1.421968	12.551017	120.743887
First Kernel	0.088236	0.214918	1.445057
Second Kernel	0.075942	0.183097	1.219344
Third Kernel	0.070847	0.133038	0.752951
	N=4000		
Serial Code	5.698898	50.223114	485.151113
First Kernel	0.210893	0.705533	5.537593
Second Kernel	0.202289	0.620481	4.668682
Third Kernel	0.182259	0.423681	2.840024

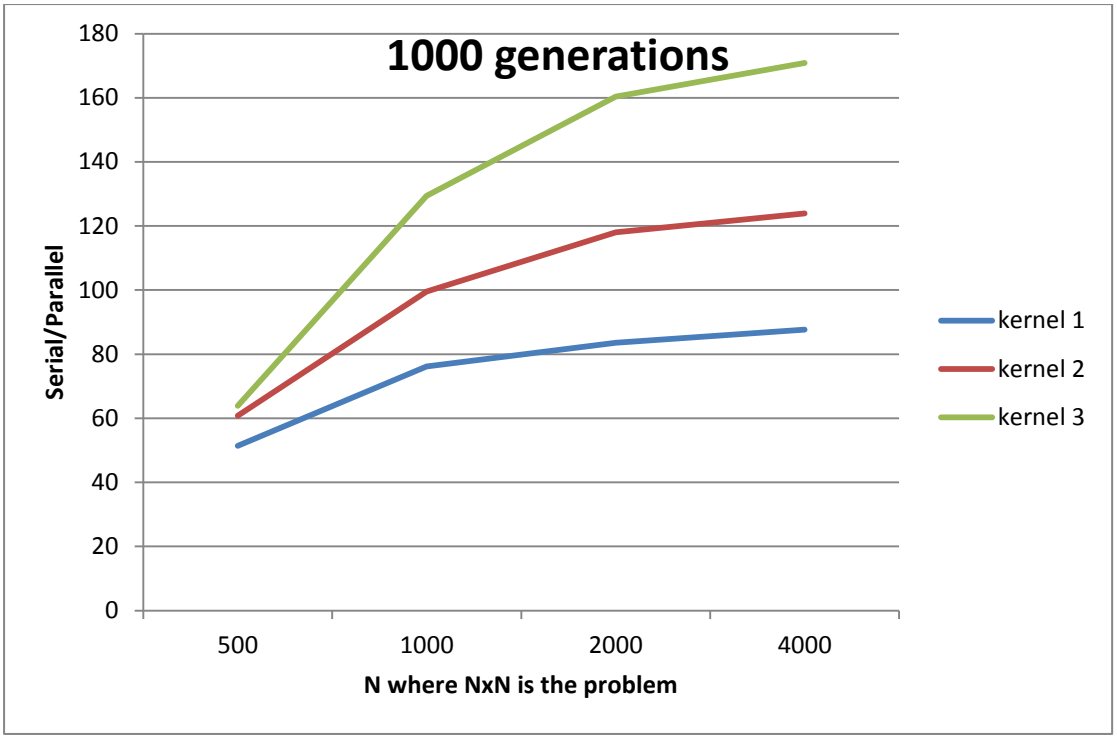
Για μικρά προβλήματα και 10 γενιές βλέπουμε ότι ο σειριακός και οι παράλληλοι κώδικες είναι στην ίδια τάξη μεγέθους.Αυτό οφείλεται στην επικοινωνία CPU με GPU.Για την ακρίβεια χρησιμοποιούμε 2 φορές την συνάρτηση cudaMemCpy() η οποία προσθέτει αρκετό χρόνο στον χρόνο εκτέλεσης.

Γραφικές παραστάσεις



Όπως παρατηρήσαμε και παραπάνω,για 10 επαναλήψεις οι παράλληλες υλοποιήσεις δεν είναι και τόσο καλές.Επίσης δεν υπάρχει εμφανής διαφορά μεταξύ των υπολοποιήσεων.





Για περισσότερες γενεές η διαφορά γίνεται πιο ορατή καθώς η αντιγραφή δεδομένων παίζει μικρό ρόλο σε σχέση με τις πράξεις.

Η επαλήθευση του αποτελέσματος έγινε με την εντολή **diff** συγκρίνοντας τα αρχεία εξόδου των πυρήνων με αυτό του σειριακού κώδικα.