ΕΡΓΑΣΙΑ ΣΤΑ ΠΑΡΑΛΛΗΛΑ ΣΥΣΤΗΜΑΤΑ

Game Of Life

Ηλιας Μεντζελος – 1115201400106
Ευαγγελος Πεκος – 1115201400157

Στον παρακατω πινακα βρισκονται οι μετρησεις του MPI

MPI					
N	120x120	240x240	360x360	480x480	600x600
1	1.91 secs	14.41 secs	30.78 secs	46.42 secs	54.16 secs
4	17.42 secs	41.48 secs	76.65 secs	62.06 secs	109.5 secs
9	4.24 secs	17.28 secs	24.69 secs	30.61 secs	97.26 secs
16	11.47 secs	24.29 secs	65.14 secs	101.96 secs	146.55 secs

Στον παρακατω πινακα βρισκονται οι μετρησεις MPI + OPEN mp

OpenMp					
n	120x120	240x240	360x360	480x480	600x600
1	3.66 secs	12.96 secs	25.82 secs	32.94 secs	67.14
4	34.55 secs	25.36secs	47.05 secs	66.37 secs	84.1 secs
9					
16					

Παρατηρηση 1 : Δυστυχως λογω της καταστασης των μηχανηματων της σχολης οι μετρησεις που πηραμε δεν ηταν ακριβεις ακομα και μετα απο πολλες προσπαθιες.Συνεπως ο υπολογισμος speed-up και efficiency δεν ηταν δυνατος επειδη οι χρονοι που περναμε απο δυο διαδοχικες εκτελεσεις (με ιδια ιnput data) μπορει να είχαν τεραστιες διαφορες.

Εδω βρισκονται οι μετρησεις cuda

CUDA	120*120	240*240	360*360	480*480	600*600
256 threads	0.0004687	0.00048614	0.00096202	0.00162482	0.00254107
1024 threads	0.00014186	0.00042915	0.00091887	0.00161600	0.00244713

Το CUDA το τρεξαμε σε δικο μας μηχανημα οπου εκει εγιναν και οι μετρησεις.

Το speedup σε σχεση με την σειριακη υλοποιηση ειναι

Speedup	120x120	240x240	360x360	480x480	600x600
	13.57	33,02	33.97	39.98	21.69

Καποια Συμπερασματα:

Παρατηρουμε πως στις υλοποιησεις MPI και OpenMP οταν εχουμε μικρα δεδομενα οι υλοποιησεις ειναι πιο αργες απο το σειριακο λογω της καθηστερησης που δημιουργει η επικοινωνια.Απο την αλλη σε μεγαλα δεδομενα οι υλοποιησεις ειναι πολυ πιο αποδοτικες επειδη και επισης παρατηρουμε οτι οσο περισσοτερες διεργασιες τοσο πιο γρηγορη γινεται η εκτελεση.

Λιγα λογια για το CUDA

Για το cuda εργαστηκαμε ως εξης:
Καταρχας χρησιμοποιουμε 2 kernels rows και colums για να συμπληρωσουν τις παραπανω γραμμες και στηλες της περιμετρου στην μνημη της GPU ή αλλιως device memory.Επειτα εχουμε τον βασικο kernel GameOfLife που θα υλοποιησει ουσιαστικα το παιχνιδι. Χρησιμοποιουμε δισδιαστατα μεγεθη για το μεγεθος του μπλοκ και του πινακα (blocksize kai gridsize). Δηλωνουμε τους πινακες που χρειαζονται για την υλοποιηση, δηλαδη 3 πινακες με προθεμα device που βρισκονται στην GPU και 3 με προθεμα host που τρεχει στον CPU. Με τους πινακες αρχικα αρχικοποιουμε τυχαια τον πληθυσμο σε εναν host πινακα και τον αντιγραφουμε στον device πινακα ετσι ωστε να επεξεργαστει απο τον Kernel. Στην συνεχεια ο kernel στον δευτερο device πινακα οριζει την δευτερη γενια συμφωνα με τους κανονες του παιχνιδου. Γινεται swap ετσι ωστε η επομενη γενια να αντιγραφει στον αρχικο πινακα και να συνεχισει η επαναληψη. Στην main συναρτηση υπολογιζεται ο νεος πινακας για καθε γενια (συνολο γενεων 200). Δηλωνουμε στον προεπεξεργαστη το BLOCKSIZE οπου το χρησιμοποιυμε για την υπολογισμο των Threads και των blocks. Στην περιπτωση που τρεχουμε με 256 threads/block οριζουμε το BLOCKSIZE 16 και αντιστοιχα 32 για 1024 threads.
Τελος τους kernels rows-colums τους καλω με αριθμο thread ισο με το BLOCKSIZE και μπλοκς ισα με το ταβανι της διαιρεσης (αριθμος στηλων ή γραμμων)/BLOCKSIZE. Πρακτικα επειδη αυτοι οι δυο kernels ασχολουνται με μια διασταση του πινακα (γραμμες και στηλες αντιστοιχα) ο αριθμος μπλοκς και threads ισουται με την τετραγωνικη ριζα των αντιστοιχων τιμων του kernel GameOfLife.Να προσθεσουμε οτι μεσα στην κεντρικη επαναληψη του παιχνιδιου ελεγχουμε τους δυο πινακες σειριακα (παλιας και νεασ γενιας) ετσι ωστε σε περιπτωση που δεν εγινε καποια αλλαγη να σταματησει η επαναληψη. Αυτο επιτυγχανεται αντιγραφοντας τους δυο device πινακες δυο host οι οποιοι δεσμευονται στην αρχη και αποδεσμευονται στο τελος της

επαναληψης.Επισης για τις μετρησεις χρονου χρησιμοποιησαμε την βιβλιθηκη timer.h που ειναι στην ιστοσελιδα http://www.cs.usfca.edu/~peter/cs625/code/timer.h.