ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Παράλληλα Συστήματα

Υλοποιήσεις: MPI, MPI/OPENMP, CUDA

Θεματολογία: Συνέλιξη

Παιδάκης Θεοδόσης - 1115201500118

Παπαχρήστου Δημήτρης - 1115201500124

1) ΟΔΗΓΊΕΣ ΜΕΤΑΓΛΩΤΙΣΣΗΣ ΚΑΙ ΕΚΤΈΛΕΣΗΣ

MPI version:

```
$ mpicc -o mpi_convolution mpi_convolution.c
$ mpiexec -n [processors] ./mpi_convolution [image.raw] [width] [height] [repetitions] [grey,rgb]
```

Παράδειγμα: `$ mpiexec -n 4 ./mpi_convolution waterfall_1920_2520.raw 1920 2520 40 rgb`

OPENMP version:

```
$ mpicc -openmp -o mpi_convolution mpi_convolution.c
$ Ίδιο με του mpi
```

CUDA version:

```
$ make
$ ./cuda_convolution [image.raw] [width] [height] [repetitions] [grey,rgb]
```

2) ΠΑΡΑΔΟΧΕΣ

- Πρώτο μέλημα μας είναι να σιγουρευτούμε ότι οι τιμές που δόθηκαν στην γραμμή εκτέλεσης ως παράμετροι είναι σωστές. Έπειτα παίρνουμε τις τιμές των μεταβλητών που μας ενδιαφέρουν και τις κάνουμε broadcast(BCAST)

- Ανάλογα τον αριθμό των διαθέσιμων διεργασιών που έχουμε, χωρίζουμε την εικόνα σε μέρη (RowsDivision()) και η κάθε διεργασία λαμβάνει και επεξεργάζεται το κομμάτι που της αναλογεί. Αυτό επιταχύνει σημαντικά το πρόγραμμα, μιας και η κάθε διεργασία διαβαζει αποκλειστικά το κομμάτι εικόνας που της αναλογεί.

- Η διαδικασία της συνέλιξης όπως είναι φυσικό, διαφέρει ως ένα βαθμό, ανάλογα με το αν η εικόνα είναι grey ή rgb.

- Γίνεται χρήση μονοδιάστατου πίνακα, καθώς αποδείχθηκε αποτελεσματικότερος. Ακόμα, με αυτλην την υλοποίηση η διαμέριση διευκολύνεται σημαντικά.

3) ΕΠΙΚΟΙΝΩΝΙΑ ΜΕΤΑΞΥ ΔΙΕΡΓΑΣΙΩΝ

Οι διεργασίες μοιράζονται έτσι στο πλέγμα, ώστε να είναι η μία δίπλα στην άλλη με αύξουσα σειρά. Αυτό διευκολύνει στην ανεύρεση των γειτονικών διεργασιών.
Ιδιάιτερη περίπτωση αποτελούν οι διεργασίες που βρίσκονται στην περιφέρεια του πλέγματος. Σε αυτήν την περίπτωση γειτονικές διεργασίες θεωρούνται αυτές που βρίσκονται στην κατάλληλη εκτός ορίων απέναντι πλευρά.

Προχωρώντας τώρα σε θέματα κώδικα σαν επιλογή έχουμε αποφασίσει να χρησιμοποιήσουμε nonblocking αποστολές και λήψεις μηνυμάτων μπορώντας με αυτόν τον τρόπο να κάνουμε συνέλιξη για τα κεντρικά κομμάτια του πίνακα (inner data) και αφήνοντας για αργότερα τον υπολογισμό των περιφερειακών κομματιών(outer and corner data). Αυτό δίνει την δυνατότητα σε μια διεργασία να ασχολείται με μια εργασία όσο περιμένει να έρθουν τα γειτονικά κελιά του πίνακα. Όταν ολοκληρωθεί η εργασία αυτή αν έχουν έρθει τα δεδομένα που περιμένει προχωράει στην εκτέλεση των απαιτούμενων νέων εργασιών αλλιώς εξακολουθεί να περιμένει την λήψη των απαιτούμενων δεδομένων.

Για την αποστολή και την λήψη, αποφασίσαμε να γίνουν με την βοήθεια Datatypes δεδομένων όπου ήταν εφικτό.

Γενικό μοτίβο υλοποίησης:

```
for loop
        ISend
        IRecv
        Inner Data Computations
        Wait(RRecv)
        Outer Data Computations
        Wait(RSend)
end for
```

OpenMP:

Σκοπός εδώ είναι κομμάτια που εκτελούνται τοπικά, να παραλληλοποιηθούν με τη χρήση thread μέσω του OpenMP, ώστε να βελτιώνεται η ήδη υλοποιημένη εκδοχή του MPI.
Συγκεκριμένα στην άσκηση, αλλάχθηκε το σημείο, στο οποίο διατρέχουμε με επανάληψη τον τοπικό πίνακα διεργασίας.

CUDA:

Το πρόγραμμα βασίστηκε στις προηγούμενες υλοποιήσεις (MPI).

Περιληπτική περιγραφή:

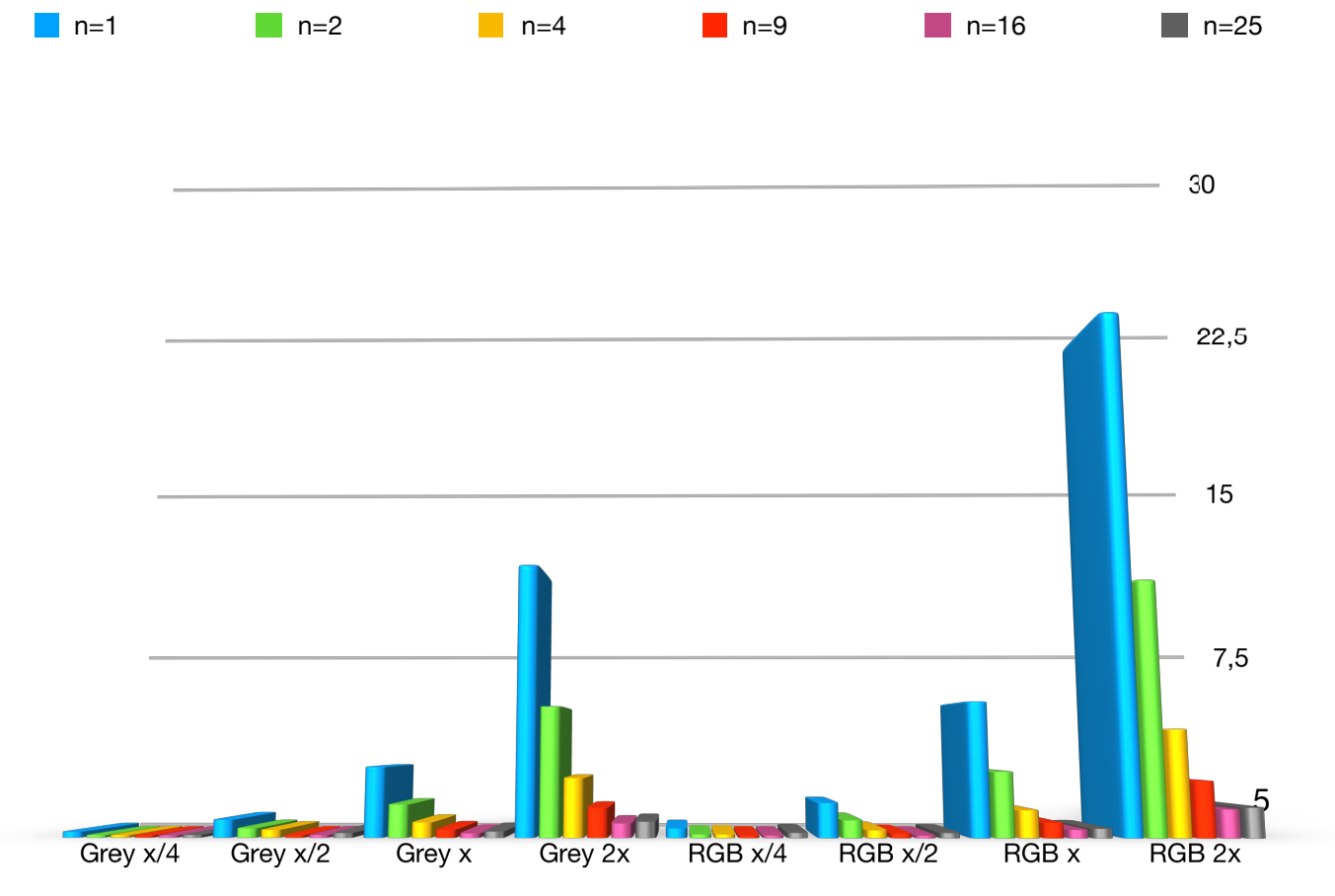
- Δεσμεύσεις χώρου στην κάρτα γραφικών.
- Αντιγραφές πινάκων (host -> device και device -> host)
- Δημιουργία και κλήση της συνάρτησης __global__
- Απελευθέρωση δεδομένων (free)

Τέλος, οι μετρήσεις έγιναν σε υπολογιστή με λογισμικό Manjaro Linux και κάρτα γραφικών MSI GEFORCE GTX 970 4GB.

Πίνακας χρόνου MPI

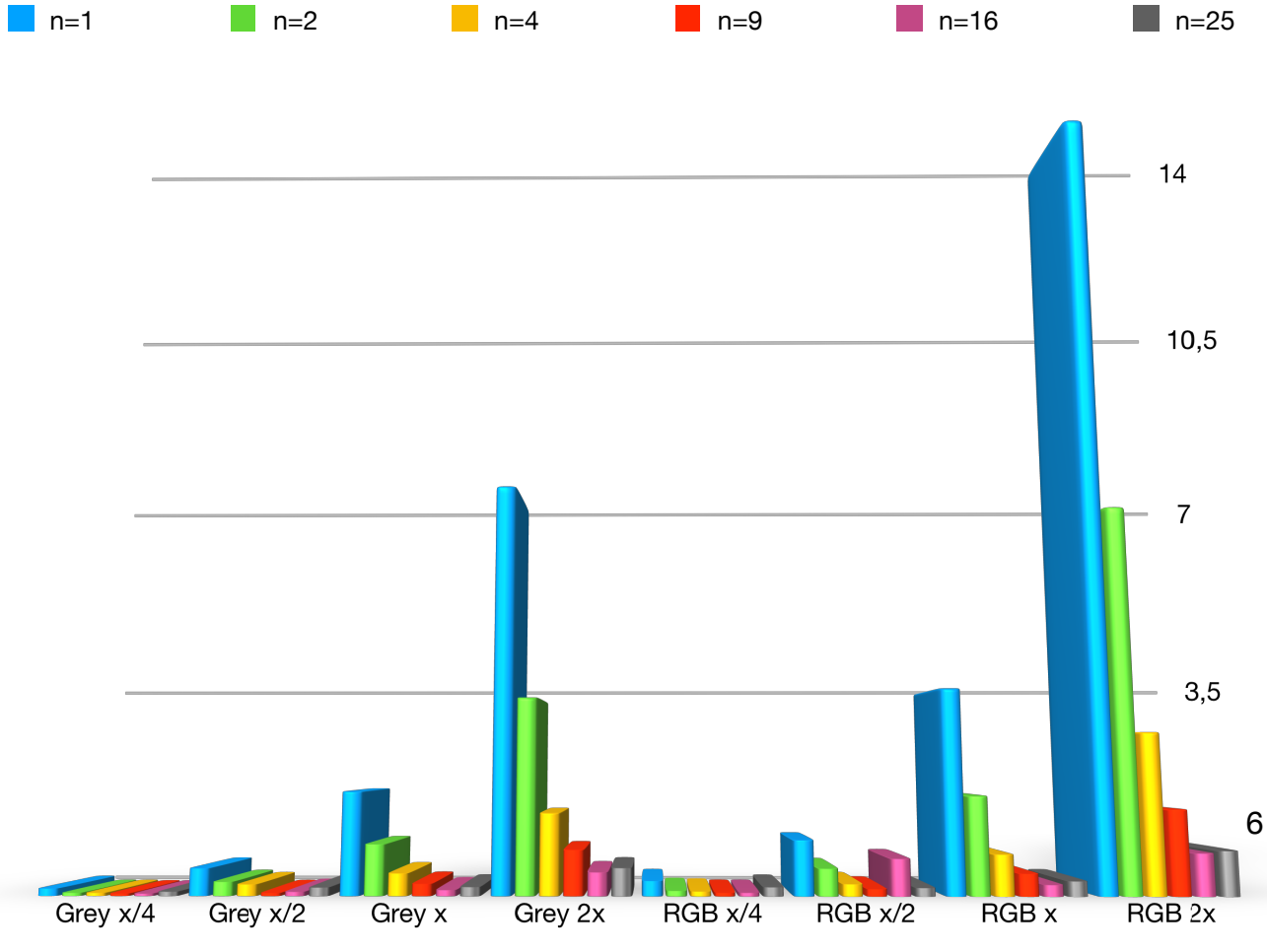
(20 repetitions)

Image Size/ Processes	1	2	4	9	16	25
Grey 1920*630 (x/4)	0.22	0.09	0.09	0.08	0.08	0.1
Grey 1920*1260 (x/2)	0.69	0.36	0.3	0.09	0.1	0.18
Grey 1920*2520(x)	2.73	1.28	0.59	0.31	0.17	0.24
Grey 1920*5040 (2x)	10.71	5.09	2.3	1.15	0.56	0.63
RGB 1920*630 (x/4)	0.37	0.14	0.14	0.1	0.1	0.2
RGB 1920*1260 (x/2)	1.35	0.68	0.3	0.17	0.1	0.2
RGB 1920*2520(x)	5.27	2.55	1.07	0.59	0.32	0.37
RGB 1920*5040 (2x)	21.09	10.08	4.2	2.19	1.11	1.18



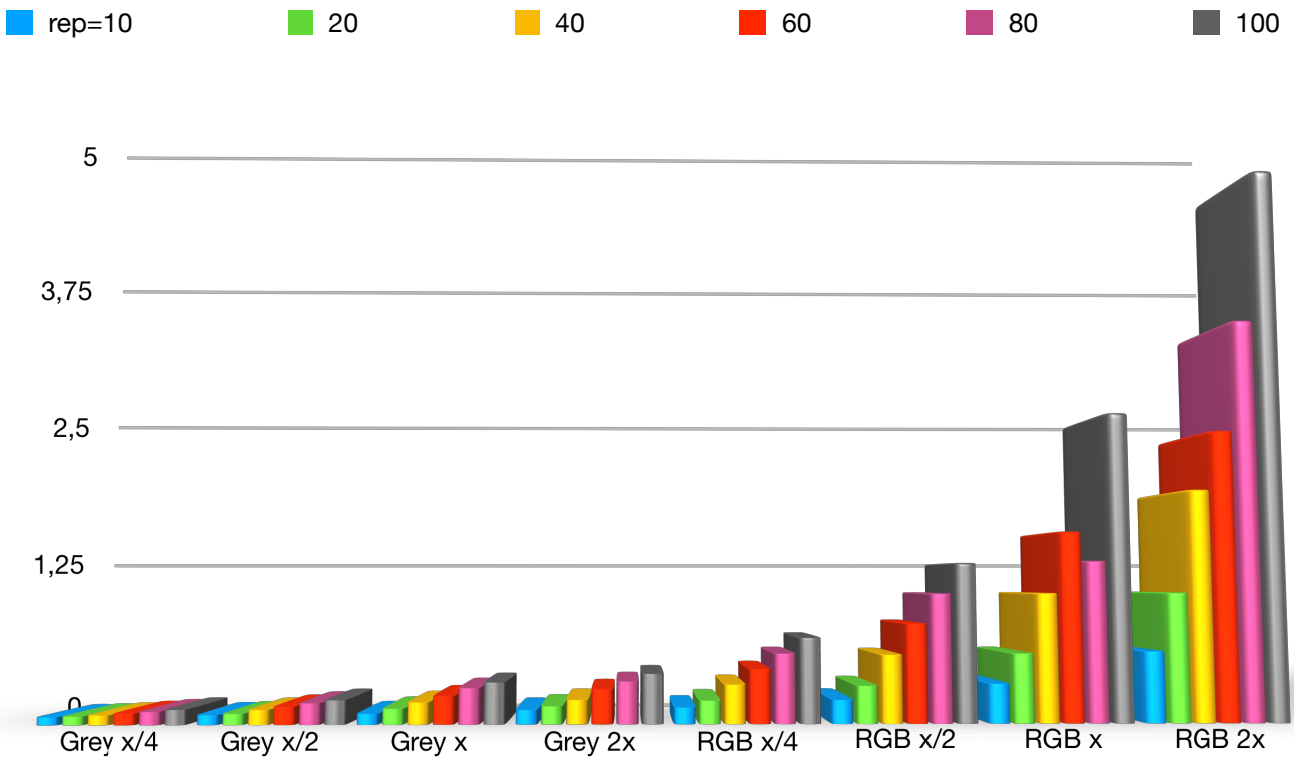
Πίνακας χρόνου OpenMP
(20 repetitions)

Image Size/ Processes	1	2	4	9	16	25
Grey 1920*630 (x/4)	0.12	0.05	0.05	0.04	0.04	0.07
Grey 1920*1260 (x/2)	0.45	0.23	0.19	0.05	0.07	0.14
Grey 1920*2520(x)	1.7	0.84	0.37	0.2	0.1	0.15
Grey 1920*5040 (2x)	6.87	3.26	1.35	0.75	0.39	0.46
RGB 1920*630 (x/4)	0.25	0.09	0.08	0.06	0.06	0.15
RGB 1920*1260 (x/2)	0.91	0.45	0.2	0.12	0.61	0.14
RGB 1920*2520(x)	3.41	1.63	0.68	0.37	0.19	0.25
RGB 1920*5040 (2x)	13.47	6.49	2.68	1.41	0.7	0.74



Πίνακας χρόνου CUDA

Image Size/ Repetitions	10	20	40	60	80	100
Grey 1920*630 (x/4)	0.062	0.068	0.076	0.089	0.103	0.118
Grey 1920*1260 (x/2)	0.077	0.086	0.116	0.141	0.168	0.191
Grey 1920*2520(x)	0.085	0.123	0.172	0.221	0.283	0.326
Grey 1920*5040 (2x)	0.113	0.141	0.189	0.272	0.332	0.39
RGB 1920*630 (x/4)	0.13	0.183	0.307	0.426	0.547	0.67
RGB 1920*1260 (x/2)	0.188	0.296	0.537	0.78	1.015	1.254
RGB 1920*2520(x)	0.311	0.546	1.017	1.503	1.272	2.448
RGB 1920*5040 (2x)	0.562	1.021	1.837	2.308	3.2	4.428



Συμπεράσματα

- Παρατηρώντας το MPI και OpenMP, φαίνεται ότι σε λίγα δεδομένα αρκούν λίγες διεργασίες, γιατι όσο μεγαλώνει ο αριθμός τους, αυξάνει το κόστος κατασκευής τους.

- Παρατηρώντας το CUDA, βλέπουμε πως οι χρόνοι αυξάνονται για τα πιο μεγάλα μεγέθη. Αντίθετα, σε χαμηλά μεγέθη πληρώνουμε το κόστος δημιουργίας του thread, μιας και το πρόγραμμα μπορεί να δουλέψει πιο βέλτιστα με λιγότερα.