

ΠΑΡΑΛΛΗΛΑ ΣΥΣΤΗΜΑΤΑ

ΕΡΓΑΣΙΑ ParGraphs

Ομάδα:

- **Ον/μο:** Κομίνη Συμέλα Φωτεινή **ΑΜ:** 1115201400072
- **Ον/μο:** Χατζηδάκης Ραφαήλ **ΑΜ:** 1115201400248

Υλοποίηση σε MPI

Οδηγίες μεταγλώττισης και εκτέλεσης:

Για την μεταγλώττιση του Cuda:

- ◆ \$ make, για την δημιουργία του εκτελέσιμου **mpi_ex**

Για την εκτέλεση:

- ◆ \$ mpirexec -np <proc_num> ./mpi_ex <image> <image_type> <height> <width> <loops>

, όπου:

- <**image**>, η εικόνα που θέλουμε να φιλτράρουμε
- <**image_type**>, αν η εικόνα είναι ασπρόμαυρη πληκτρολογούμε GREY, αν είναι έγχρωμη πληκτρολογούμε RGB (αυτές είναι και οι μόνες 2 αποδεκτές τιμές)
- <**height**>, <**width**>, το ύψος και το πλάτος της εικόνας αντίστοιχα
- <**loops**>, ο αριθμός των επαναλήψεων που επιθυμούμε

Επεξήγηση Κώδικα:

- Το πρόγραμμα ξεκινάει την εκτέλεσή του με τον έλεγχο των arguments που πληκτρολόγησε ο χρήστης. Αν βρεθεί έστω και ένα λάθος τότε το πρόγραμμα ενημερώνει τον χρήστη με το αντίστοιχο error και τερματίζει. Αυτοί οι έλεγχοι είναι:

- 1) Αριθμός των arguments.
- 2) Αριθμός των loops(>0)
- 3) Αριθμός των processes(πρέπει να είναι δύναμη του 2)

4)Τύπος εικόνας

5)Αν μπορεί να διαιρεθεί σε blocks ανάλογα με το πόσα processes έχουμε

6)Αν η εικόνα υπάρχει

7)και αν οι διαστάσεις που δόθηκαν είναι σωστές.

-Μετά ανοίγει η εικόνα δίνοντας σε κάθε process έναν δείκτη στην αρχη του αρχείου και καθε process χρησιμοποιεί τον δείκτη αυτόν για να διαβάσει μόνο το κομμάτι του block του.

-Αφού ετοιμαστούν τα blocks, κάθε process ελέγχει για γείτονες στους οποίους στέλνει το κατάλληλο κομμάτι του block. Όταν τα processes έχουν στείλει και έχουν πάρει τα κομμάτια των blocks αρχίζει ο αλγόριθμος για το convolution. Αυτό γίνεται συνέχεια μέχρι να τελειώσουν τα loops ή μεχρι να να μην μπορεί να αλλάξει άλλο η εικόνα.

-Στο τέλος δημιουργείται μια καινούργια εικόνα γράφοντας σε ένα αρχείο που έχει δεσμευτεί, με το μέγεθος της αρχικής εικόνας, με τον ίδιο τρόπο που διαβάστηκε.

Μετρήσεις:

Οι μετρήσεις έγιναν στα linux της σχολής.

Χρόνος Εκτέλεσης σε seconds					
Μέγεθος εικόνας / Αριθμός Processes	4	9	16	25	36
GREY 1920x630	2.097	2.453	-	3.022	3.228
GREY 1920x1260	3.601	3.769	5.576	5.757	-
GREY 1920x2520	9.923	10.009	9.232	13.131	14.658
GREY 1920x5040	15.939	14.388	21.06	22.103	27.645
RGB 1920x630	2.883	2.791	-	3.491	3.707
RGB 1920x1260	6.458	6	7.525	7.876	7.772
RGB 1920x2520	9.185	10.273	12.963	16.297	14.544
RGB 1920x5040	17.205	17.575	18.205	19.982	-

Υλοποίηση σε MPI-OPENMP

Οδηγίες μεταγλώττισης και εκτέλεσης:

Για την μεταγλώττιση του Cuda:

- \$ make, για την δημιουργία του εκτελέσιμου **mpi_openmp_ex**

Για την εκτέλεση:

- \$ mpirexec -np <proc_num> ./mpi_ex <image> <image_type> <height> <width> <loops>

Επεξήγηση Κώδικα:

-Χρησιμοποείται ο ίδιος κώδικας με το MPI, με την διαφορά του [**#pragma omp parallel for private (i,j) collapse (2)**] για την επανάληψη στον αλγόριθμο του convolution.

Μετρήσεις:

Οι μετρήσεις έγιναν στα linux της σχολής.

Χρόνος Εκτέλεσης σε seconds					
Μέγεθος εικόνας / Αριθμός Processes	4	9	16	25	36
GREY 1920x630	0.42	0.983	-	2.652	3.601
GREY 1920x1260	0.738	1.286	2.2	3.088	-
GREY 1920x2520	1.288	2.217	3.334	4.332	5.155
GREY 1920x5040	2.592	4.146	5.867	6.783	8.043
RGB 1920x630	1.06	2.177	-	3.786	4.741
RGB 1920x1260	2.031	3.152	4.283	5.384	6.868
RGB 1920x2520	4.007	5.513	7.831	8.961	10.31
RGB 1920x5040	8.144	14.611	17.743	19.003	20.729

Υλοποίηση σε CUDA

Οδηγίες μεταγλώττισης και εκτέλεσης:

Για την μεταγλώττιση του Cuda:

- ◆ \$ make, για την δημιουργία του εκτελέσιμου **cuda_ex**

Για την εκτέλεση:

- ◆ \$./cuda_ex <image> <image_type> <height> <width> <loops> <blocksize>

, όπου:

- <**blocksize**>, το μέγεθος των blocks, στα οποία θα σπάσει η εικόνα
- Τα υπόλοιπα είναι ίδια με του mri.

Επεξήγηση Κώδικα:

- Το πρόγραμμα ζεκινάει την εκτέλεσή του με τον έλεγχο των arguments που πληκτρολόγησε ο χρήστης. Αν βρεθεί έστω και ένα λάθος τότε το πρόγραμμα ενημερώνει τον χρήστη με το αντίστοιχο error και τερματίζει.
- Αν τα arguments είναι σωστά, το πρόγραμμα ζεκινά να μετρά τον χρόνο εκτέλεσης του, ανοίγει το αρχείο της εικόνας και την διαβάζει.
- Στη συνέχεια καλεί την cuda συνάρτηση filtering, όπου δεσμεύεται χώρος στην GPU για τους 2 πίνακες start και end, στους οποίους θα αποθηκεύεται κάθε φορά το αποτέλεσμα κάθε επανάληψης.
- Σε κάθε επανάληψη, ανάλογα με τον τύπο της εικόνας, δημιουργούνται τα κατάλληλα blocks και με την αντίστοιχη συνάρτηση “kernel_grey” ή “kernel_rgb” γίνεται το φιλτράρισμα των pixel της εικόνας με την χρήση threads.

- Μόλις τελειώσει η διαδικασία, αντιμεταθέτονται οι πίνακες start και end και ξεκινάει η επόμενη επανάληψη.
- Ανάλογα με τον αριθμό των επαναλήψεων, επιστρέφεται και ο αντίστοιχος πίνακας (start/end) στον host.
- Αφού γίνουν όλες οι επαναλήψεις, δημιουργείται το αρχείο “filtered.raw” μέσα στον φάκελο του εκτελέσιμου, και αντιγράφεται εκεί η φιλτραρισμένη εικόνα.
- Τέλος, υπολογίζεται ο χρόνος εκτέλεσης του προγράμματος και εκτυπώνεται μαζί με τα arguments που εισήγαγε ο χρήστης στην αρχή του.

Μετρήσεις:

Οι μετρήσεις έγιναν σε υπολογιστή με NVIDIA GeForce 840M κάρτα γραφικών.

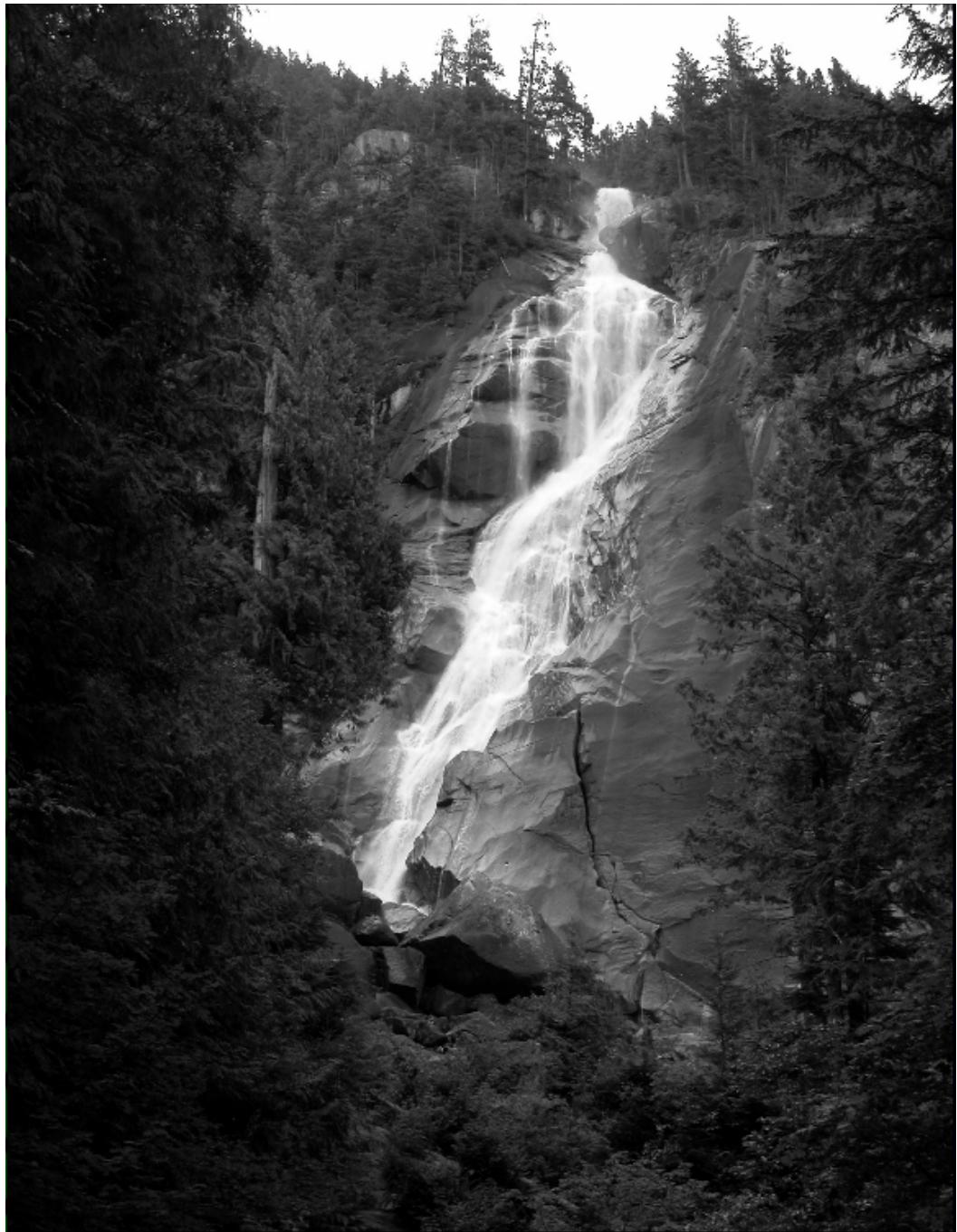
GeForce 840M, Blocksize = 9, Χρόνος Εκτέλεσης σε seconds					
Μέγεθος εικόνας / Αριθμός Επαναλήψεων	10	30	50	70	90
GREY 1920x630	0.107	0.167	0.191	0.222	0.266
GREY 1920x1260	0.143	0.241	0.287	0.375	0.467
GREY 1920x2520	0.185	0.383	0.53	0.714	0.934
GREY 1920x5040	0.28	0.663	1.08	1.454	1.833
RGB 1920x630	0.291	0.608	0.965	1.347	1.679
RGB 1920x1260	0.435	1.153	1.865	2.609	3.298
RGB 1920x2520	0.814	2.278	3.734	5.229	6.7
RGB 1920x5040	1.597	4.59	7.542	10.474	13.429

Παρατηρούμε ότι ο χρόνος αυξάνεται όσο αυξάνεται το μέγεθος της εικόνας.

Αποτελέσματα:

GREY

Αρχική εικόνα



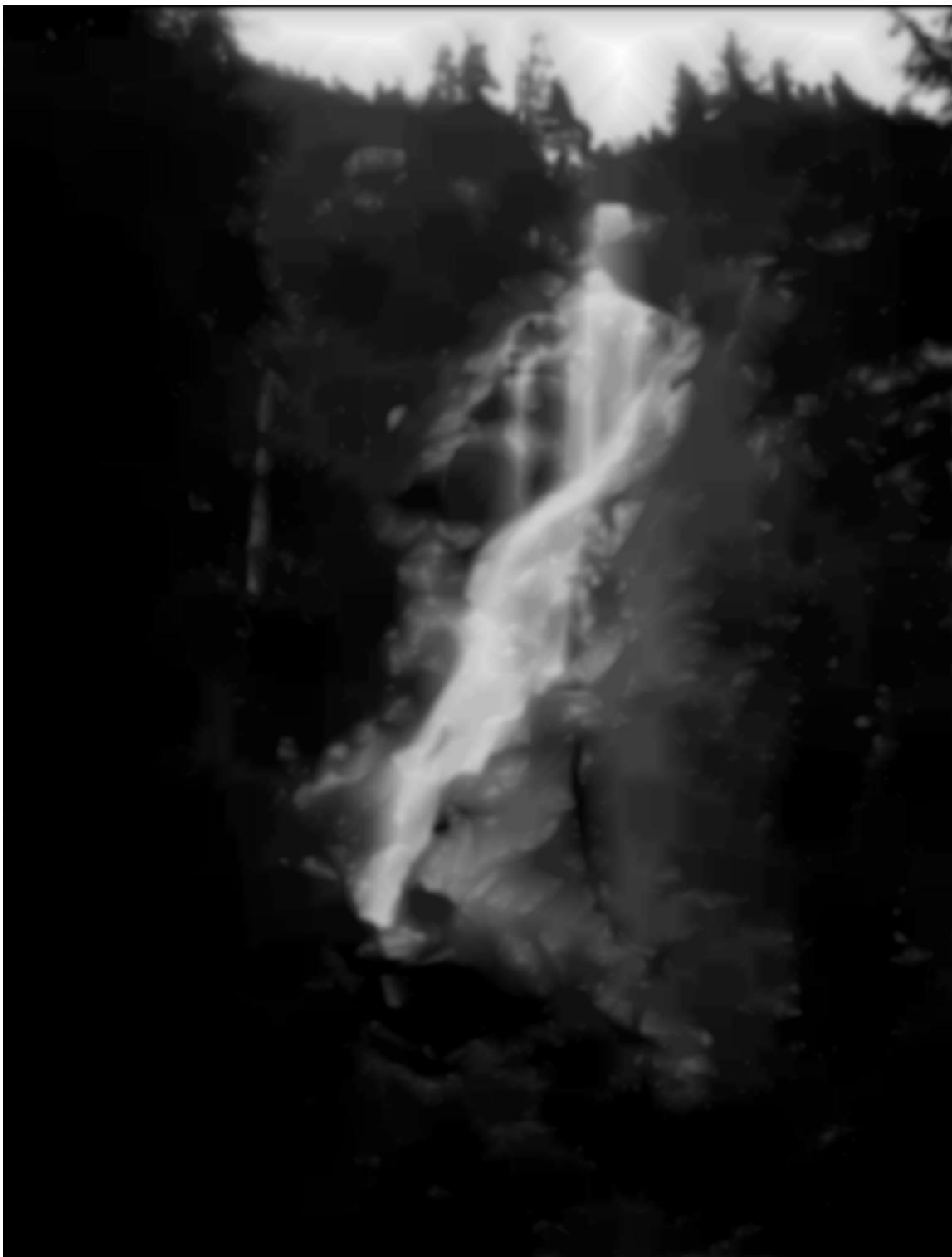
10 Επαναλήψεις



50 Επαναλήψεις



90 Επαναλήψεις



RGB

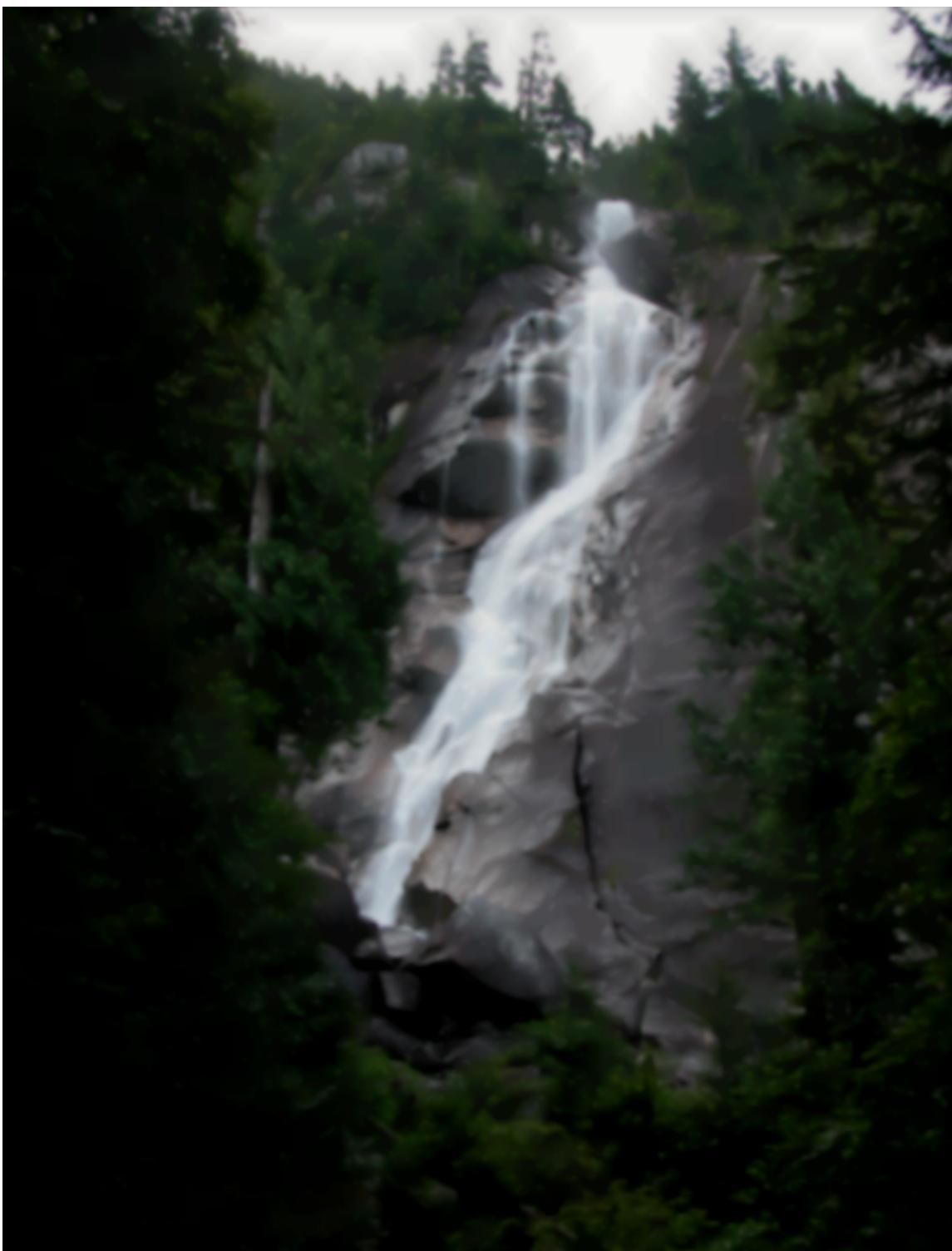
Αρχική εικόνα



10 Επαναλήψεις



50 Επαναλήψεις



90 Επαναλήψεις

