ΕΙΡΗΝΗ ΤΖΙΜΑ P3220201

Άσκηση 1

Ερώτημα α.

Η υλοποίηση του συγκεκριμένου μέρους της άσκησης στηρίχθηκε στην ιδέα της ισόποσης κατανομής του φόρτου εργασίας, όχι μόνο σε ένα νήμα, όπως συμβαίνει στην σειριακή εκτέλεση, αλλά σε 2,4 και τέλος σε 8 νήματα. Πιο αναλυτικά, έχει δημιουργηθεί μία συνάρτηση, η blur_worker(...), την οποία, όλα τα νήματα χρησιμοποιούν με διαφορετικές παραμέτρους που ορίζουν το μέρος της εικόνας που έχει αναλάβει το κάθε νήμα να θολώσει. Η blur_worker(...), εφαρμόζει την θόλωση σε κάθε pixel ξεχωριστά, για το κομμάτι της εικόνας που κάθε νήμα έχει αναλάβει να θολώσει. Στη συνέχεια, η κύρια συνάρτηση είναι η gaussian_blur_parallel(...), η οποία δέχεται ως είσοδο το filepath της εικόνας και δημιουργεί Ν νήματα τα οποία καλούν την blur_worker(), για διαφορετικό εύρος γραμμών της εικόνας. Ο διαχωρισμός αυτός βασίστηκε στον αριθμό των νημάτων και στο ύψος της εικόνας, χρησιμοποιώντας τον τύπο rows_per_thread = height / number_of_threads.
Ο τρόπος αυτός επιλέχθηκε διότι η θόλωση των pixels είναι διαδικασία ανεξάρτητη επομένως δεν δημιουργούνταν προβλήματα συγχρονισμού ή κοινής πρόσβασης σε μεταβλητές, γεγονός που καθιστά την παραλληλοποίηση πιο απλή και αποδοτική.

Σύγκριση Εκτέλεσης με 2 Threads:

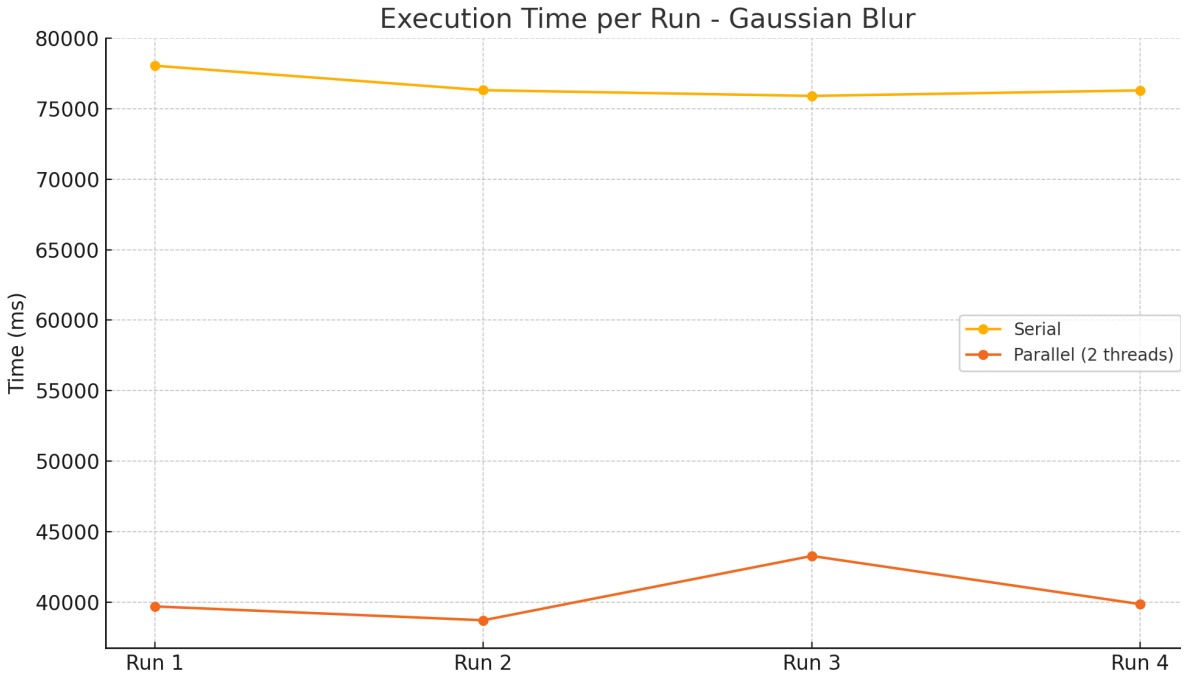
Για καλύτερη σύγκριση ανάμεσα στην σειριακή και την παράλληλη εκτέλεση, έχουν γίνει 4 επαναλήψεις για κάθε περίπτωση. Στη συνέχεια, υπολογίζεται ο μέσος όρος των χρόνων, ώστε να υπάρχει μια πιο αντικειμενική εικόνα της απόδοσης κάθε μεθόδου. Όπως φαίνεται από τον πίνακα καθώς επίσης και το διάγραμμα, η παράλληλη εκτέλεση παρουσιάζει σημαντικά μικρότερο χρόνο εκτέλεσης σε σχέση με την σειριακή.

Εκτέλεση	Serial (ms)	Parallel (2 threads) (ms)
Run 1	78036	39716
Run 2	76303	38734
Run 3	75895	43291
Run 4	76287	39877

Μέσοι όροι:

- Serial:
  (78036 + 76303 + 75895 + 76287) / 4 = 76630 ms

- Parallel (2 threads):
  (39716 + 38734 + 43291 + 39877) / 4 = 40354 ms



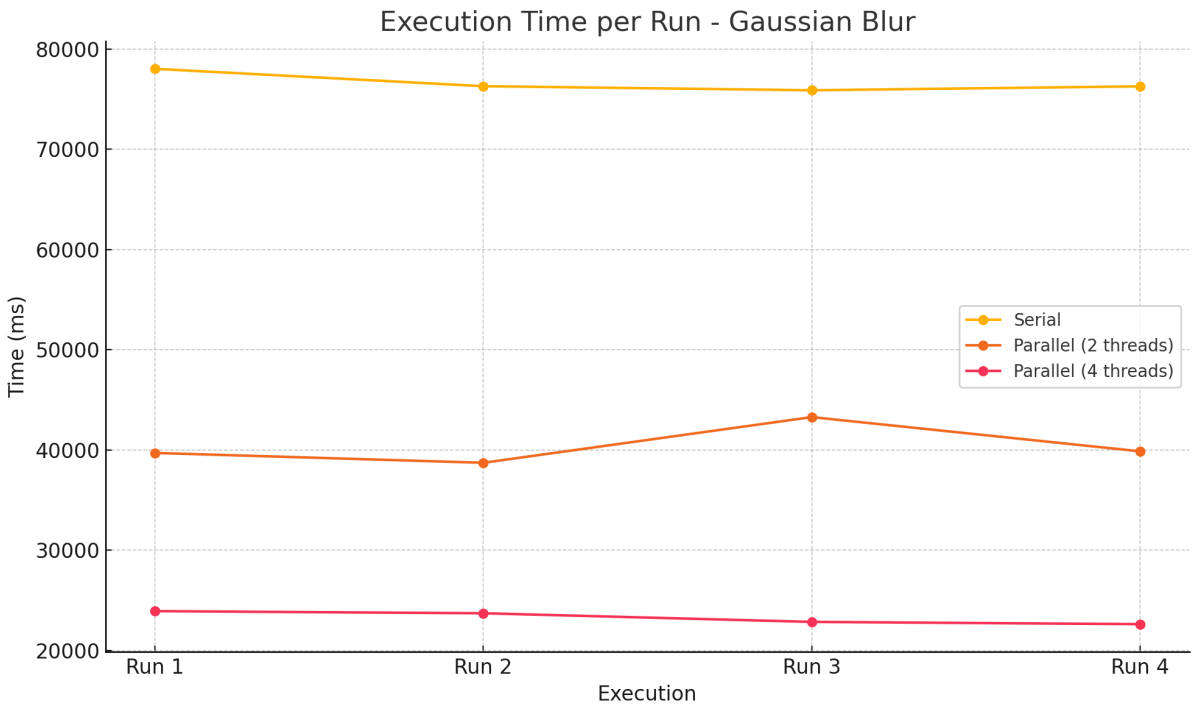
Η παράλληλη εκτέλεση με 2 threads είναι αρκετά ταχύτερη από τη σειριακή, με μέσο χρόνο περίπου 40.000 ms έναντι 76.000 ms. Η μείωση αυτή δείχνει πως ο καταμερισμός εργασίας σε νήματα μπορεί να επιταχύνει σημαντικά την εκτέλεση.

Σύγκριση Εκτέλεσης με 4 Threads:

ομοίως κάνουμε 4 μετρήσεις για πιο αξιόπιστα αποτελέσματα

Εκτέλεση	Serial (ms)	Parallel (2 threads) (ms)	Parallel (4 threads) (ms)
Run 1	78036	39716	23936
Run 2	76303	38734	23722
Run 3	75895	43291	22861
Run 4	76287	39877	22637

Μέσος όρος (4 threads): (23936+23722+22861+22637)/4=23289 ms



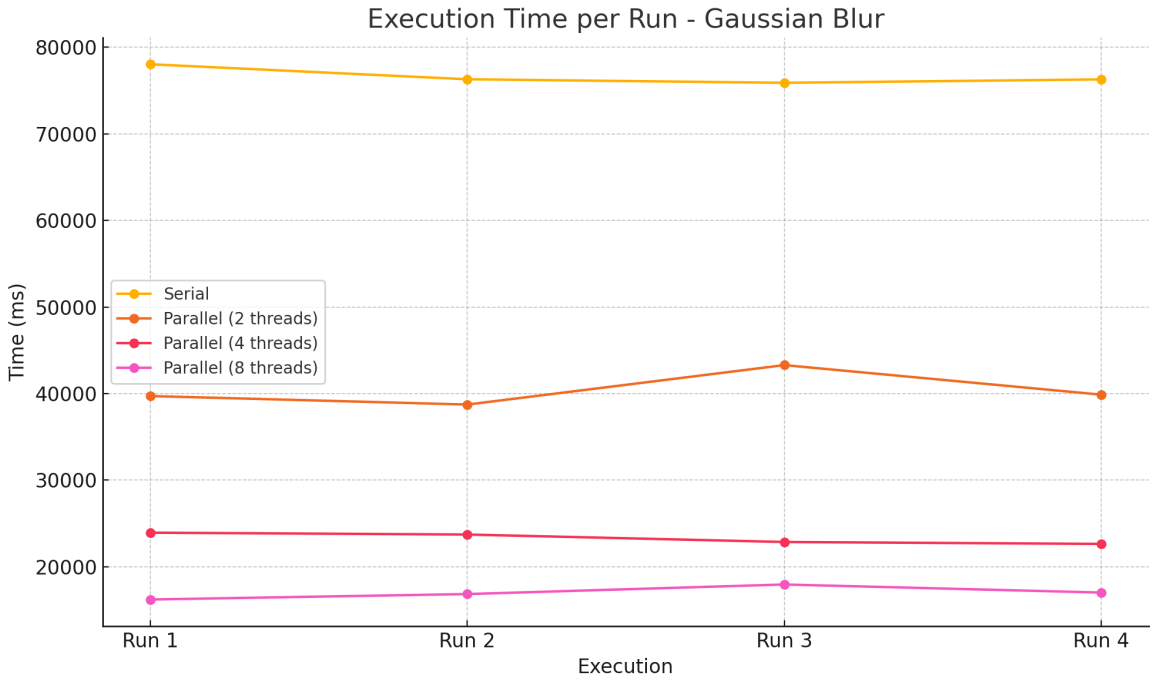
Κατά την εκτέλεση των τριών πειραμάτων, παρατηρείται ότι όσο αυξάνεται ο αριθμός των threads, μειώνεται σημαντικά ο συνολικός χρόνος εκτέλεσης της λειτουργίας Gaussian Blur. Η σειριακή εκτέλεση παρουσιάζει τον υψηλότερο χρόνο με μέσο όρο περίπου 76.630 ms, ενώ με την παράλληλη εκτέλεση σε 2 threads ο χρόνος μειώνεται σχεδόν στο μισό (40.354 ms). Η περαιτέρω αύξηση σε 4 threads οδηγεί σε ακόμα καλύτερη απόδοση, με μέσο χρόνο 23.289 ms. Αυτή η σταδιακή βελτίωση δείχνει την αποτελεσματικότητα της παραλληλοποίησης. Παρόλο που η επιτάχυνση δεν είναι απόλυτα γραμμική λόγω του κόστους συγχρονισμού και επικοινωνίας μεταξύ των νημάτων η διαφορά είναι ξεκάθαρη τόσο στους αριθμούς όσο και στο γράφημα.

Σύγκριση Εκτέλεσης με 8 Threads:

Εκτέλεση	Serial (ms)	Parallel (2 threads) (ms)	Parallel (4 threads) (ms)	Parallel (8 threads) (ms)
Run 1	78036	39716	23936	16219
Run 2	76303	38734	23722	16847

Run 3	75895	43291	22861	17952
Run 4	76287	39877	22637	17020

Μέσος όρος (8 threads): (16219+16847+17952+17020)/4 = 17009 ms



Με την αύξηση των threads σε 8, παρατηρείται ακόμα μεγαλύτερη μείωση στον χρόνο εκτέλεσης, με μέσο όρο μόλις 17.009 ms. Σε σύγκριση με τη σειριακή εκτέλεση (76.630 ms), η διαφορά είναι αρκετά αισθητή. Η βελτίωση σε σχέση με τα 4 threads είναι υπαρκτή αλλά μικρότερη, κάτι που δείχνει ότι καθώς αυξάνεται ο αριθμός των threads, το κέρδος από την παραλληλοποίηση φθίνει.

Ερωτημα β.

Για το δεύτερο μέρος της άσκησης, μας ζητείται να χρησιμοποιήσουμε 4 νήματα για να υπολογίσουμε τις μέγιστες τιμές ανά κανάλι, να κανονικοποιήσουμε τα pixels και να εφαρμόσουμε θόλωση στον οριζόντιο και κάθετο άξονα, αποθηκεύοντας τις ενδιάμεσες και τελικές εικόνες. Όλα αυτά υλοποιούνται στη συνάρτηση gaussian_blur_separate_parallel(const char* filename). Μέσα σε αυτή, δημιουργείται μία lambda συνάρτηση η thread_work την οποία εκτελούν τα νήματα, και χρησιμοποιείται std::barrier για συγχρονισμό στα κρίσιμα σημεία. Η διαδικασία χωρίζεται σε 4 στάδια: Στο πρώτο στάδιο, κάθε νήμα υπολογίζει τις μέγιστες τιμές για το δικό του τμήμα της εικόνας και

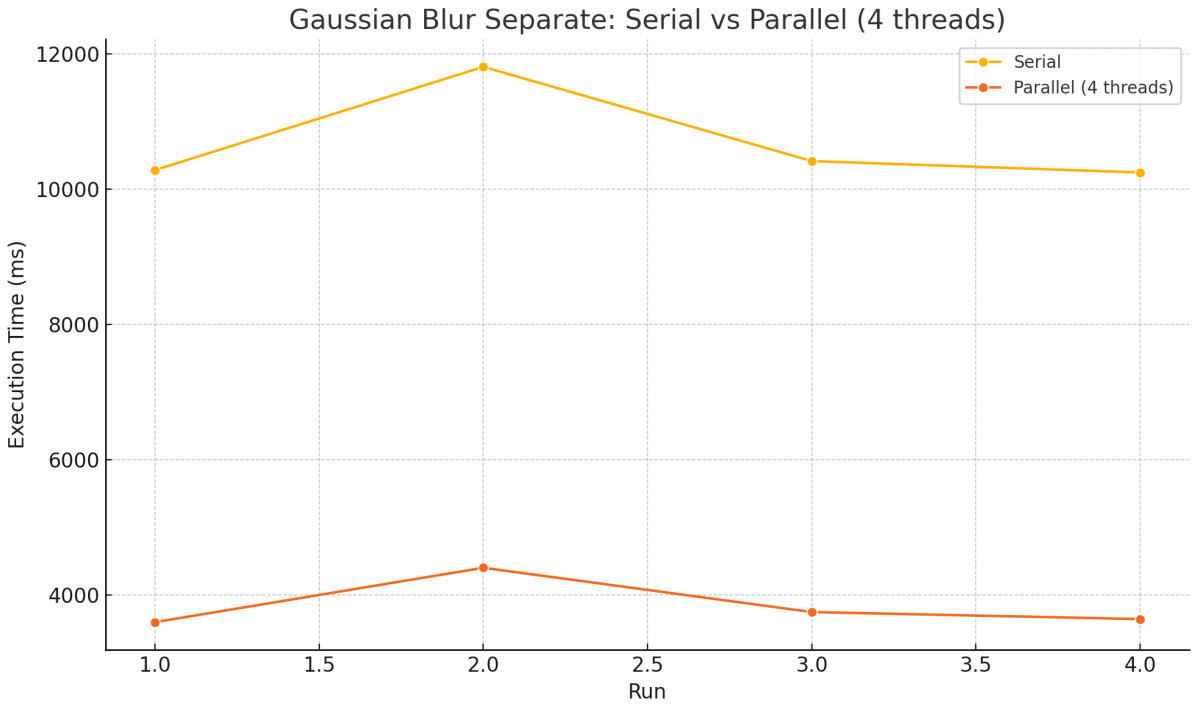
ενημερώνει τις global max τιμές με χρήση atomic. Στη συνέχεια, με τη χρήση του barrier, όλα τα νήματα συγχρονίζονται πριν προχωρήσουν στο επόμενο βήμα. Στο δεύτερο στάδιο γίνεται η κανονικοποίηση των pixels με βάση τα max ανά κανάλι. Ακολουθεί ξανά συγχρονισμός, και ένα νήμα αποθηκεύει την εικόνα. Το τρίτο στάδιο περιλαμβάνει την οριζόντια θόλωση, και το τέταρτο την κάθετη θόλωση. Και στα δύο στάδια, η αποθήκευση της εικόνας γίνεται από ένα μόνο νήμα. Η χρήση του barrier εξασφαλίζει ότι όλα τα νήματα βρίσκονται στο ίδιο στάδιο πριν προχωρήσουν, χωρίς την πολυπλοκότητα άλλων μεθόδων συγχρονισμού.

Εκτέλεση	Serial (ms)	Parallel (4 threads) (ms)
Run 1	10279	3594
Run 2	11806	4398
Run 3	10412	3743
Run 4	10243	3639

Μέσοι όροι:

- Serial:
  (10279 + 11806 + 10412 + 10243) / 4 =10685 ms

- Parallel (4 threads):
  (3594 + 4398 + 3743 + 3639) / 4 = 3843.5 ms



Από τα αποτελέσματα του ερωτήματος β, βλέπουμε ξεκάθαρα ότι η υλοποίηση με 4 threads που είναι παράλληλη είναι πιο αποδοτική από τη σειριακή. Ο μέσος χρόνος εκτέλεσης για τη σειριακή είναι περίπου 10685ms, ενώ για την παράλληλη είναι 3843.5ms. Αυτό δείχνει ότι με την χρήση πολλαπλών threads και τον διαμερισμό της διαδικασίας σε αυτά, επιτυγχάνεται καλύτερη εκμετάλλευση των πόρων του συστήματος και μειώνεται ο συνολικός χρόνος εκτέλεσης. Επομένως, η προσέγγιση με threads είναι πιο αποδοτική για την επεξεργασία εικόνας Gaussian Blur.

Συμπεράσματα:

Συμπερασματικά λοιπόν, από όλα τα πειράματα, γίνεται φανερό ότι η χρήση παραλληλισμού βελτιώνει σημαντικά την απόδοση της επεξεργασίας εικόνας μέσω Gaussian Blur. Όσο αυξάνονται τα threads, οι χρόνοι εκτέλεσης μειώνονται, δείχνοντας έτσι ότι ο υπολογισμός γίνεται ταχύτερος από την ταυτόχρονη εκτέλεση σε πολλούς πυρήνες. Η μεγαλύτερη επιτάχυνση παρατηρήθηκε όταν φτάσαμε τα 8 threads, όπου ο μέσος χρόνος μειώθηκε αρκετά σε σύγκριση με τον σειριακό. Επιπλέον, και στο ερώτημα β, η οργάνωση της διαδικασίας σε ξεχωριστά threads για κάθε στάδιο είχε ορατά θετικά αποτελέσματα. Συμπερασματικά, ο παραλληλισμός όχι μόνο μειώνει τον χρόνο εκτέλεσης, αλλά είναι και ουσιαστικά απαραίτητος για πιο αποδοτική και γρήγορη επεξεργασία εικόνας.