Παράλληλα Συστήματα
2017-2018

Σωτήρης Παπαδιαμάντης 1115201300132
Αλέξανδρος Λαποκωνσταντάκης 1115201200088

1)Εισαγωγή

Η εργασία ασχολείται με την υλοποίηση συνέλιξης σε εικόνα με χρήση φίλτρου-kernel διαστάσεων 3x3, συγκεκριμένα για την επίτευξη του εφέ Gaussian Blur 3x3 στην εικόνα. Η εικόνα τόσο στην είσοδο όσο και στην έξοδο έχει τη μορφή 2D array. Η υλοποίηση έγινε σειριακά, με MPI και με hybrid MPI-OpenMP.

2)Δομή Project

Για το compile χρησιμοποιείται makefile
Ενδεικτικές εντολές εκτέλεσης:

```
mpirun -np 1 convolution -r 3000 -c 4000
mpirun -np 4 convolution -r 1520 -c 1920 -i input.txt
```

Όπου -r για rows, -c για columns, -i για input αρχείο-εικόνα και -np για χρήση cores
Αν δε δοθεί -i παράμετρος η εικόνα δημιουργείται με χρήση rand().
Η σειριακή υλοποίηση επιλέγεται αν δοθεί -np 1 στο MPI. Το -np 1 στο MPI-openMP αναμενόμενα σημαίνει τη χρήση threads σε ένα μόνο core.

Φάκελος MPI

- main.c
- convolution.c
- convolution.h
- mpiFunctions.c
- mpiFunctions.h

Φάκελος openMP

- main.c
- convolution.c
- convolution.h
- mpiFunctions.c
- mpiFunctions.h

3) Στοιχεία Παράλληλου Προγραμματισμού

Τοπολογία

Χρησιμοποιείται καρτεσιανή τοπολογία, και την ορίζουμε με την MPI_Cart_Create(). Είναι τετραγωνική καθώς έχει διαστάσεις cores*cores, όπου cores^2 είναι ο αριθμός των πυρήνων που δόθηκε σαν επιλογή κατά την εκτέλεση του προγράμματος. Η κάθε διάσταση των blocks-υποπινάκων που προκύπτουν απ’ την ανάθεση των τμημάτων της αρχικής εικόνας στις διεργασίες είναι το προϊόν της ακέραιας διαίρεσης με τις αντίστοιχες διαστάσεις του αρχικού πίνακα με τον αριθμό cores, πχ subRows = rows/cores, άρα όλοι οι υποπίνακες θα έχουν τον ίδιο αριθμό γραμμών και στηλών μεταξύ τους, αλλά δεν είναι απαραίτητα τετραγωνικοί.

Στο mpiFunctions.h έχουμε ορίσει ακέραιους αριθμούς για τον προσδιορισμό της θέσης των γειτονικών διεργασιών, ενώ η τοπολογία είναι περιοδική για τον υπολογισμό των ακραίων στοιχείων του πίνακα.

Έτσι, η κάθε διεργασία παίρνει:

1. την τελευταία γραμμή της πάνω διεργασίας (N)
2. το στοιχείο [n-2][1] της πάνω δεξιά διεργασίας (NE)
3. την πρώτη στήλη της δεξιά διεργασίας (E)
4. το στοιχείο [1][1] της κάτω δεξιά διεργασίας (SE)
5. την πρώτη γραμμή της κάτω διεργασίας (S)
6. το στοιχείο [1][n-2] της κάτω αριστερά διεργασίας (SW)
7. την τελευταία στήλη της αριστερά διεργασίας (W)
8. το στοιχείο [n-2][n-2] της πάνω αριστερά διεργασίας (NW)

Επικοινωνία μεταξύ διεργασιών

Ο αρχικός πίνακας(I1) διαβάζεται σειριακά απο τη master διεργασία και στη συνέχεια διαμοιράζεται με MPI_Scatterv στις υπόλοιπες, κρατώντας τον 1ο υποπίνακα για υπολογισμό από την ίδια.
Μετά των ορισμό των υποπινάκων και το διαμερισμό του αρχικού, τα στοιχεία τους μεταφέρονται με memcpy (λόγω ταχύτητας) σε ένα νέο με 2 εξωτερικές γραμμές και 2 εξωτερικές στήλες παραπάνω, μία σε κάθε πλευρά για να κρατήσει το halo των γειτονικών διεργασιών.

Έτσι ορίζονται στο convolution.c αρχείο 2 πίνακες για τα requests, οι MPI_Request sendReq[8] και MPI_Request recvReq[8].

Χρησιμοποιούνται οι MPI_Isend και MPI_Irecv για non-blocking αποστολή και λήψη μηνυμάτων απο τις γειτονικές διαδικασίες, μέσω των sendHalo και recvHalo συναρτήσεων που υλοποιήσαμε αντίστοιχα. Η findNeighProcs χρησιμοποιείται για να βρει τις γειτονικές διεργασίες αυτή που την καλεί.

Στην MPI υλοποίηση χρησιμοποιείται η MPI_Testany μέσα σε while loop και τα εξωτερικά στοιχεία του κάθε υποπίνακα υπολογίζονται ανάλογα με τα μέρη του halo που έρχονται πρώτα, αντι να περιμένει να τα λάβει όλα.

Στην MPI-openMP περιμένει μέχρι να έχει ολόκληρο το halo για να είναι δυνατή η χρήση for loop και να καταμεριστεί η εργασία σε threads.

Μετά το πέρας των πολλαπλών συνελίξεων (χρησιμοποιείται ένα for loop) στους υποπίνακες τα δεδομένα συγκεντρώνονται στο συνολικό πίνακα συνέλιξης (I0) με την MPI_Gatherv.

Χρήση ειδικών τύπων δεδομένων

Ορίζονται τα rows και τα columns των υποπινάκων με τις συναρτήσεις:

```
MPI_Type_contiguous(pCols[rank]-2, MPI_INT, &ROW);
MPI_Type_commit(&ROW);
MPI_Type_vector(pRows[rank]-2, 1, pCols[rank], MPI_INT, &COLUMN);
MPI_Type_commit(&COLUMN);
```

Οι πίνακες και υποπίνακες έχουν γίνει allocate έτσι ώστε να είναι contiguous για να είναι δυνατόν να οριστούν αυτοι οι 2 τύποι γραμμής και στήλης.
Το -2 χρησιμοποιείται για να μη συμπεριλαμβάνονται τα στοιχεία του halo κατά την αποστολή/λήψη των αντίστοιχων στοιχείων.

Το πρόγραμμα το τρέξαμε στα linux της σχολής.
Οι παρακάτω μετρήσεις, τόσο για MPI όσο και για openMP γίνονται πάνω στο χρόνο που χρειάζεται να εκτελεστεί συνέλιξη 50 φορές πάνω στον πίνακα χωρίς τη χρήση reduce για διακοπή του loop νωρίτερα. Υπολογίζεται μόνο ο χρόνος του loop, αποκλείοντας τις αρχικοποιήσεις, διάβασμα του αρχείου και γράψιμο output πριν και μετά. Χρησιμοποιήθηκαν τυχαία δημιουργημένες εικόνες με τη rand και σα να ήταν ασπρόμαυρες(1 τιμή για κάθε pixel). Για αυτές τις μετρήσεις επιλέχτηκαν επίσης να είναι τετραγωνικοί.

---

4)Μετρήσεις MPI

Μετρήσεις Χρόνου για MPI

για σειριακή (1) και χρησιμοποιώντας 4, 9 και 16 cores.

Μέγεθος Πίνακα	**1**
240x240	0.210733
480x480	0.856951
960x960	3.124611
1920x1920	13.645787
3840x3840	54.102629

Μέγεθος Πίνακα	**4**	**9**	**16**
240x240	0.127346	0.071985	0.066116
480x480	0.426391	0.408190	0.396401
960x960	1.748335	1.484732	1.623780
1920x1920	7.584622	4.679200	5.749656
3840x3840	26.254301	22.814410	19.841500

Μελέτη Επιτάχυνσης για MPI

Χρησιμοποιώ τον τύπο S = Tserial / Tparallel

Μέγεθος Πίνακα	**4**	**9**	**16**
240x240	1.6548	2.9274	3.1873
480x480	2.0097	2.0993	2.1618
960x960	1.7871	2.1044	1.9242
1920x1920	1.7991	2.9162	2.3733
3840x3840	2.0607	2.3714	2.7267

Μελέτη Απόδοσης για MPI
Χρησιμοποιώ τον τύπο της απόδοσης E = S/p, όπου S η επιτάχυνση όπως υπολογίστηκε και p ο αριθμός των διεργασιών.

Μέγεθος Πίνακα	4	9	16
240x240	0.4137	0.3252	0.1992
480x480	0.5024	0.2332	0.1351
960x960	0.4467	0.2338	0.1202
1920x1920	0.4497	0.3240	0.1483
3840x3840	0.5151	0.2634	0.1704

Μελέτη Κλιμάκωσης για MPI

Παρατηρούμε ότι δεν υπάρχουν μεγάλες αποκλίσεις στην επιτάχυνση για κάθε μέγεθος πίνακα όταν χρησιμοποιούνται περισσότεροι από ένας πυρήνες, αν και αναμενόμενα όσο περισσότεροι χρησιμοποιούνται αυξάνεται. Γενικά το πρόγραμμα δεν παρουσιάζει καλή κλιμάκωση, οι αυξήσεις στην απόδοση είναι μικρές, Υποθέτουμε οφείλεται στο ότι ενώ οι υπολογισμοί γίνονται γρηγορότερα όσο αυξάνονται οι πυρήνες, αυξάνεται και ο χρόνος που παίρνει η μεταφορά στοιχείων με memcpy όσο μεγαλώνει ο πίνακας, πράγμα που ήταν μονόδρομος με τον τρόπο που δομήσαμε τον κώδικα (δε διαβάζουν παράλληλα οι διεργασίες το input, αλλά γίνεται scatter εκ των υστέρων). Έτσι, ενώ για μεγαλύτερα μεγέθη πίνακα περιμέναμε να αυξάνεται κατά πολύ η απόδοση του προγράμματος όσο χρησιμοποιεί μεγαλύτερο αριθμό πυρήνων, κάτι τέτοιο δε συμβαίνει.

5)Σχεδιασμός OpenMP

Έχει πραγματοποιηθεί υλοποίηση του OpenMP χρησιμοποιώντας την MPI_THREAD_MULTIPLE εκδοχή του ορίσματος support της MPI_Init_thread().

-Εντολή #pragma omp parallel num_threads(4)

Προσθήκη της παραπάνω εντολής ακριβώς πριν τον βρόχο επανάληψης for για παραλληλοποίηση των γύρων συνέλιξης, σ' αυτήν την περίπτωση χρησιμοποιούνται 4 threads.

- Εντολή: #pragma omp single

Χρήση αυτής της εντολής στα τμήματα που αντιστρέφονται οι πίνακες input (I1) και output (I0), στην ανταλλαγή halo και στα reset του πίνακα στόχου έτσι ώστε να πραγματοποιούνται μόνο από ένα thread.

- Εντολή: #pragma omp for

Χρήση αυτής της εντολής στο εσωτερικό loop του υπολογισμού συνέλιξης, ουσιαστικά ένα row διαμοιράζεται στα threads για να υπολογιστεί.

Μετρήσεις Χρόνου OpenMP

Για τις παρακάτω μετρήσεις έχουν γίνει οι ίδιες συμβάσεις που ισχύουν και για τις μετρήσεις στο MPI. Χρησιμοποιήθηκαν 4 threads.

Μέγεθος Πίνακα	**4**	**9**	**16**
240x240	0.133677	0.075087	0.102665
480x480	0.524675	0.405106	0.348718
960x960	1.695185	1.520612	1.466927
1920x1920	6.332038	5.739987	5.763907
3840x3840	28.390878	24.434258	23.374202

Μελέτη Επιτάχυνσης OpenMP

Χρησιμοποιώντας τον τύπο S = Tserial/Tparallel

Μέγεθος Πίνακα	**4**	**9**	**16**
240x240	1.5764	2.8065	2.0526
480x480	1.6332	2.1153	2.4574
960x960	1.8432	2.0548	2.1300
1920x1920	2.1550	2.3773	2.3674
3840x3840	1.9056	2.2142	2.3146

Μελέτη Απόδοσης OpenMP

Χρησιμοποιώντας τον τύπο E = S/p

Μέγεθος Πίνακα	**4**	**9**	**16**
240x240	0.3941	0.3118	0.1282
480x480	0.4083	0.2350	0.1535
960x960	0.4608	0.2283	0.1331
1920x1920	0.5387	0.2641	0.1479
3840x3840	0.4764	0.2460	0.1446

Μελέτη Κλιμάκωσης OpenMP

Για μικρά μεγέθη πίνακα (240 x 240 & 480 x 480) παρατηρούμε πως με την αύξηση των χρησιμοποιούμενων διεργασιών, η αποδοτικότητα φθίνει. Η κλιμάκωση είναι κακή και σχεδόν μηδενική, αν όχι αρνητική σε μερικές περιπτώσεις.

Σύγκριση με απλό MPI

Σε γενικές γραμμές η επιτάχυνση και η απόδοση κινούνται στα ίδια επίπεδα στο openMP και στο απλό MPI αλλά φαίνεται ότι η απόδοση του openMP πέφτει όσο αυξάνεται το πλήθος των πυρήνων και περνάει μπροστά, έστω και για ελάχιστο σε ορισμένες περιπτώσεις το MPI.