Παραλληλοποίηση συσσωρευτικού αλγορίθμου ανάλυσης οπτικού πεδίου

Ζέλιος Ανδρέας, Γραμμένος Αχιλλέας
ece01326@uowm.gr, ece01312@uowm.gr

Περίληψη - **Στα πλαίσια του μαθήματος Συστήματα Παράλληλης και Κατανεμημένης Επεξεργασίας του Πανεπιστημίου Δυτικής Μακεδονίας, διδαχθήκαμε τον προγραμματισμό MPI, OpenMP καθώς και τον συνδυασμό τους με σκοπό την μείωση του χρόνου εκτέλεσης σειριακών προγραμμάτων. Η εργασία αυτή αποσκοπεί στην περαιτέρω εκβάθυνση στον τομέα αυτό μέσω της παραλληλοποίησης ενός προγράμματος βασισμένο σε ένα πρόβλημα του πραγματικού κόσμου, το πρόβλημα της Ανάλυση οπτικού πεδίου. Με αφορμή έτσι την διπλωματική εργασία Ανάλυση οπτικού πεδίου με την χρήση παράλληλης επεξεργασίας, έχουμε ως στόχο την δημιουργία σειριακού κώδικα σε γλώσσα C, που εκτελεί σάρωση 360 μοιρών από ένα σημείο x, y μιας εικόνας σε Geographical Tag Image File Format και ανακαλύπτει ποια σημεία είναι εμφανή από εκεί και ποια όχι. Αυτό αποτυπώνεται σε μια εικόνα, με το κάθε σημείο της να χρωματίζεται ανάλογα με το πόσο εμφανές είναι. Στην συνέχεια θα προχωρήσουμε στην παραλληλοποίηση του κώδικα σε OpenMp, MPI και συνδυασμό τους με καταγραφή της μέτρησης χρόνου εκτέλεσης σε κάθε περίπτωση έχοντας σκοπό μια βελτίωση του χρόνου αυτού. Η μελέτη του προβλήματος θα επεκταθεί και στις παρατηρήσεις στην αποτελεσματικότητα παραλληλοποίησης, σημεία που χρειάζονται βελτίωση λόγο λαιμόκοψης καθώς και μελέτη κλιμάκωσης των αποτελεσμάτων όπως θα πραγματοποιούσε ενας προγραμματιστής παράλληλου κώδικα σε ένα σενάριο του πραγματικού κόσμου με στόχο την μελλοντική βελτίωση του κώδικα όπως μας έχει διδαχθεί στο μάθημα Συστήματα Παράλληλης και Κατανεμημένης Εργασίας.**

Λέξεις κλειδιά - Ανάλυση οπτικού πεδίου, Παράλληλη επεξεργασία, Παρεμβολή, Συντελεστής διεύθυνσης, OpenMp, MPI.

I. ΕΙΣΑΓΩΓΗ

Η ανάλυση οπτικού πεδίου αποτελεί θέμα που ενδιαφέρει πολλούς τομείς, συμπεριλαμβανομένου και των τηλεπικοινωνιών [1]. Ο σειριακός κώδικας της παρούσας εργασίας δέχεται σαν παράμετρο ζευγάρια συντεταγμένων πολλών παρατηρητών, αυτό μπορεί να εκφραστεί ως ένα αριθμό πιθανών προς τοποθέτηση κεραιών που επιθυμούμε να βρούμε το εύρος σήματός τους. Με αυτό το τρόπο μπορούμε με ένα τρόπο να μελετήσουμε τα βέλτιστα σημεία που μπορούμε να τις τοποθετήσουμε καθώς και να εξοικονομήσουμε πόρους και χρόνο για την επίτευξη του έργου. Το παρόν πρόβλημα έχει επιλυθεί με πολλούς και διάφορους τρόπους όπως τον αλγόριθμο R2, R3, Xdraw καθώς και τον Van kreveld από τον Randolph Fraklin κ.ά.[2], η παρόν υλοποίηση προσεγγίζει τους αλγόριθμους R2, R3 εκμεταλλευόμενος την γραμμική παρεμβολή [3]. Χωρίσαμε την ανάλυση της εργασίας σε τέσσερις τομείς:

- **Σειριακός Κώδικας – Μεθοδολογία**, με επεξήγηση του τρόπου λειτουργίας του κώδικα που δημιουργήσαμε.
- **Παράλληλος Κώδικας – Προβλήματα**, με ανάλυση του τρόπου προσέγγισης της παραλληλοποίησης σε OpenMp, MPI, συνδυασμού τους καθώς και τυχόν προβλήματα που αντιμετωπίσαμε.
- **Πειράματα – Αξιολόγηση απόδοσης**, με σύγκριση του σειριακού κώδικα με τις άλλες τρείς μορφές του σε μορφή που είναι παραλληλοποιήσιμη και εκτίμηση των σημείων που μπορεί να υστερούν καθώς και γραφική αναπαράσταση των αποτελεσμάτων με διαγράμματα.
- **Συμπεράσματα - Μελλοντικές βελτιώσεις**, με εκτίμηση του έργου και τρόπων εμπλουτισμού του μελλοντικά.

II. ΚΩΔΙΚΑΣ

I. Σειριακός Κώδικας – Μεθοδολογία

Ο σειριακός κώδικας αποτελείται από τρείς κύριες συναρτήσεις καθώς και πέντε βοηθητικές, επίσης δημιουργήθηκε και μια δομή struct με ένα ζεύγος συντεταγμένων x, y για τις συντεταγμένες. Η κύρια συνάρτηση main δέχεται ως παραμέτρους το όνομα του αρχείου εικόνας TIFF το ύψος των παρατηρητών, την αρχική μοίρα, την τελική μοίρα, το βήμα που γίνεται η αύξηση των μοιρών καθώς και το αρχείο που περιέχει τα ζεύγη των συντεταγμένων (μέγιστο 100) των παρατηρητών, υπάρχει η δυνατότητα να εκλεχθούν όλα τα σημεία πληκτρολογώντας -1. Αφού λάβει τις παραμέτρους δημιουργεί μια εικόνα BMP ιδίου μεγέθους με αυτή που έλαβε (το μέγεθος είναι σταθερό και 3142 επί 2254) και την χρωματίζει μαύρη αποτυπώνοντας μηδενική ορατότητα. Έπειτα εκτελεί περιστροφή με βάση τις παραμέτρους που έλαβε γύρω από τον παρατηρητή χρησιμοποιώντας τις δύο βασικές συναρτήσεις και αποτυπώνει τα αποτελέσματα ορατότητας του στην εικόνα ανοίγοντας την απόχρωση του μαύρου (ανάλογα με το πλήθος των παρατηρητών) έχοντας το άσπρο ως το σημείο που έχει πλήρη ορατότητα, το βήμα επαναλαμβάνεται έπειτα και για τους υπόλοιπους παρατηρητές. Οι πέντε βοηθητικές συναρτήσεις είναι οι:

- **float radian_from_degree(float degree)**, που μετατρέπει τις μοίρες από μορφή ακτίνια σε μοίρες.
- **float lambda_line(float degree)**, που βρίσκει τον συντελεστή διεύθυνσης με βάση την μοίρα που λαμβάνει.
- **bool valid_point_inside_picture(int x ,int y)**, για έλεγχο εάν το σημείο που εξετάζουμε βρίσκεται εντός εικόνας.
- **float getheight(void *data0 ,int x ,int y)**, για την εύρεση του ύψους του εξεταζόμενου σημείου.
- **void compute_direction(float degrees ,int *x_direction ,int *y_direction)**, για τον εντοπισμό του τεταρτημόριου που βρίσκεται το εξεταζόμενο σημείο

Οι δύο κύριες συναρτήσεις είναι οι **float compute_height_at_x_y()**, καθώς και η **void compute_height_degree()**. Η compute_height_degree() χρησιμοποιεί την compute_direction(), για να βρει το τεταρτημόριο που βρίσκεται και με βάση την compute_height_at_x_y() συνεχίζει να προχωρά στην κατεύθυνση που δόθηκε ή σταματά εάν βρεθεί ότι το ύψος που δόθηκε δεν είναι εμφανές στον παρατηρητή. Σημειώνεται ότι σε αυτή την συνάρτηση γίνεται ο χρωματισμός της εικόνας μέχρι το σημείο μηδενικής ορατότητας. Το σημείο αυτό θα είναι κρίσιμο για την παράλληλη υλοποίηση στο MPI, μιας και η συνάρτηση δέχεται ως όρισμα εικόνα BMP, ένα είδος δηλαδή δομής struct δύσκολα διαχειρίσιμο από threads.Ακολουθεί ο ψευδοκώδικας:

ΨΕΥΔΟΚΩΔΙΚΑΣ 1: ΚΥΡΙΟΣ ΣΕΙΡΙΑΚΟΣ ΑΛΓΟΡΙΘΜΟΣ

```
ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ ΒΑΣΗ ΠΑΡΑΜΕΤΡΩΝ
ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΜΑΥΡΙΣΜΑ ΤΗΣ ΕΙΚΟΝΑΣ BMP
ΓΙΑ 0 ΕΩΣ ΑΡΙΘΜΟ ΠΑΡΑΤΗΡΗΤΩΝ ΜΕ ΒΗΜΑ 1
    ΓΙΑ ΑΡΧΙΚΗ ΜΟΙΡΑ ΕΩΣ ΤΕΛΙΚΗ ΜΟΙΡΑ ΜΕ ΕΙΣΑΓΟΜΕΝΟ ΒΗΜΑ
        ΥΠΟΛΟΓΙΣΕ ΤΑ Χ, Υ ΑΝΑΛΟΓΑ ΤΟ ΤΕΤΑΡΤΗΜΟΡΙΟ
        ΕΝΟΣΩ Ο ΠΑΡΑΤΗΡΗΤΗΣ ΒΛΕΠΕΙ
            ΕΑΝ ΔΕΝ ΕΙΜΑΣΤΕ ΕΝΤΟΣ ΕΙΚΟΝΑΣ BREAK
            ΠΡΟΧΩΡΗΣΕ ΚΑΤΑ X_DIRECTION, Y_DIRECTION
            ΒΡΕΣ ΥΨΟΣ ΜΕ ΒΑΣΗ ΤΑ X_DIRECTION, Y_DIRECTION
            ΕΑΝ ΥΨΟΣ < ΤΩΡΙΝΟ ΥΨΟΣ ΠΑΡΑΤΗΡΗΤΗ
            ΕΠΕΣΤΡΕΨΕ Χ, Υ ΜΑΧ ΕΙΔΑΛΛΩΣ ΒΡΕΣ ΤΗΝ ΠΡΟΗΓΟΥΜΕΝΗ ΤΙΜΗ ΤΩΝ PIXEL ΚΑΙ ΠΡΟΣΘΕΣΕ ΤΗΝ ΤΙΜΗ ΠΟΥ ΕΧΕΙ ΑΡΧΙΚΟΠΟΙΗΘΕΙ ΣΤΗΝ MAIN
```

II. Παράλληλος Κώδικας – Προβλήματα

i. Κώδικας OpenMP

Βλέποντας τον παραπάνω ψευδοκώδικα της σειριακής υλοποίησης εντοπίζουμε δύο πιθανά σημεία εκμετάλλευσης της παραλληλοποίησης επιπέδου βρόγχου, ο πρώτος εξωτερικός βρόγχος καθώς και ο αμέσως εσωτερικός. Κρίθηκε καταλληλότερο η παραλληλοποίηση του εσωτερικού βρόγχου μιας και οι περισσότερες πράξεις γίνονται σε αυτόν. Ένα βασικό πρόβλημα στην υλοποίηση αποτέλεσε το γεγονός ότι το omp διαχειρίζεται δύσκολα δεκαδικούς αριθμούς στην εντολή omp parallel for, έτσι ήταν αναγκαίο να αλλάξουμε τις παραμέτρους με το βήμα και το εύρος που διασχίζει για να έχουν την μορφή ακεραίων. Για τον σκοπό αυτό χρησιμοποιήσαμε την μέθοδο της αναγωγής ορίζοντας δύο ακέραιους Τέλος και Αρχή βρόγχου και αλλάζοντας την προηγούμενη μεταβλητή Idegree της συνάρτησης compute_height_degree() με βάση το μετρητή βρόγχου ως εξής:

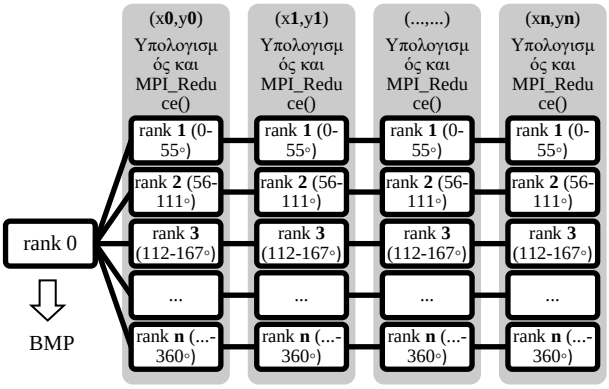
ΨΕΥΔΟΚΩΔΙΚΑΣ 2: ΜΕΘΟΔΟΣ ΑΝΑΓΩΓΗΣ ΤΟΥ ΜΕΓΕΘΟΥΣ ΒΡΟΓΧΟΥ FOR

```
ΤΕΛΟΣ ΒΡΟΓΧΟΥ ← ΜΕΓΙΣΤΕΣ ΜΟΙΡΕΣ / ΟΡΙΣΜΕΝΟ ΒΗΜΑ
ΑΡΧΗ ΒΡΟΓΧΟΥ ← ΑΡΧΙΚΗ ΜΟΙΡΑ / ΟΡΙΣΜΕΝΟ ΒΗΜΑ
ΓΙΑ ΜΕΤΡΗΤΗΣ ΒΡΟΓΧΟΥ = ΑΡΧΗ ΒΡΟΓΧΟΥ ΕΩΣ ΤΕΛΟΣ ΒΡΟΓΧΟΥ ΜΕ ΒΗΜΑ 1
    IDEGREE ← IDEGREE × ΜΕΤΡΗΤΗΣ ΒΡΟΓΧΟΥ
    COMPUTE_HEIGHT_DEGREE(…, IDEGREE, …)
```

Επόμενο βήμα ήταν προσθήκη του #pragma omp parallel με default (shared) μιας και εξ' ορισμού ο μετρητής του βρόγχου είναι private μεταβλητή. Η απόδοση του κώδικα ήταν αρκετά καλή με μεγάλη μείωση του χρόνου εκτέλεσης.

ii. Κώδικας MPI

Ο κώδικας MPI κρίθηκε πιο δύσκολος στην παραλληλοποίηση μιας και χρειάστηκαν πολύ περισσότερες αλλαγές από τον κώδικα OpenMP. Η κύρια ιδέα είναι ο master rank 0 να διαβάζει και να αρχικοποιεί τα δεδομένα, να τα στέλνει με MPI_Bcast() στα υπόλοιπα ranks και αυτά έπειτα από τον κατάλληλο διαμοιρασμό τους να χρωματίζουν την εικόνα και να ενώνουν τα αποτελέσματα τους σε μία ενιαία εικόνα. Αποδείχθηκε πως η εικόνα BMP αποτελεί ένα τύπου struct που δεν υπάρχει στην βιβλιοθήκη MPI και έτσι δεν μπορεί να σταλθεί η ληφθεί από καμία συνάρτηση του, εκτός αν δημιουργηθεί με την συνάρτηση MPI_Type_create_struct().



ΓΡΑΦΗΜΑ 1: ΥΠΟΛΟΓΙΣΜΟΣ (0-360◦), Ν ΠΑΡΑΤΗΡΗΤΕΣ ΣΕ MPI

Η παραλλαγή που εφαρμόσαμε μετέφερε τον χρωματισμό της εικόνας στο master εξαλείφοντας αυτό το πρόβλημα. Κάθε rank μετά τον υπολογισμό μεταφέρει την τιμή χρώματος (ανά τοις μοίρες του οπτικού πεδίου του παρατηρητή που έχει αναλάβει) της εικόνας σε ένα πίνακα

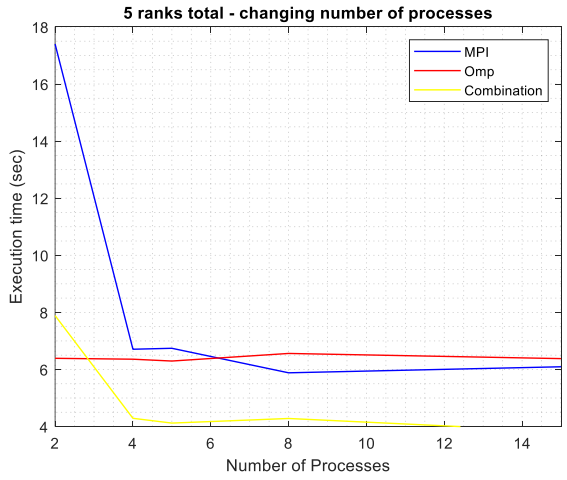
ίσου μεγέθους με αυτή, οι τιμές του χρώματος κάθε rank αθροίζονται και έπειτα αποστέλλονται με MPI_Reduce() στο master για να τις χρησιμοποιήσει στον χρωματισμό της εικόνας BMP όπως βλέπουμε και στο προηγούμενο γράφημα. Στο παράδειγμα αυτό έχουμε εισαγωγή από τον χρήστη 0 - 360◦ και η παρατηρητές, όμως αυτό αλλάζει ανάλογα το τι επιθυμεί ο χρήστης.

iii. Κώδικας MPI-OpenMp

Για τον συνδυασμό των δύο μεθόδων παραλληλοποίησης χρησιμοποιήσαμε και συνδυασμό των τεχνικών που χρειάστηκαν για τις δύο προηγούμενες υλοποιήσεις. Ο κώδικας παρέμεινε ίδιος με τον κώδικα MPI, αλλάζοντας μόνο την εσωτερική for που κάνει τον υπολογισμό του χρώματος ανά σημείο που βλέπει ο παρατηρητής κάνοντας αναγωγή για να μπορεί να το χειριστεί το OpenMp. Έτσι τα δεδομένα χωρίζονται στα threads και έπειτα το εύρος του βρόγχου κάθε thread χωρίζεται σε ακόμη περισσότερα threads κάνοντας την υλοποίηση ακόμη πιο αποδοτική.

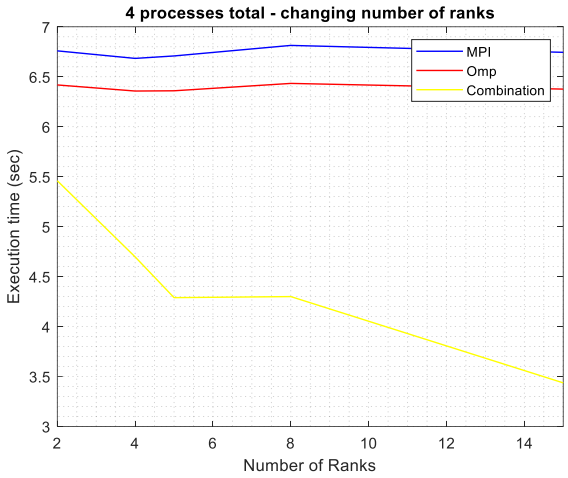
III. ΠΕΙΡΑΜΑΤΑ – ΑΞΙΟΛΟΓΗΣΗ ΑΠΟΔΟΣΗΣ

Τα πειράματα που εκτελέσαμε αφορούσαν την εναλλαγή του αριθμού των threads και των διεργασιών και είχαμε σαν μέτρο σύγκρισης τον χρόνο εκτέλεσης. Στο πρώτο πείραμα επιλέξαμε σαν σταθερά 5 threads, αλλάζοντας τον αριθμό των διεργασιών. Στο δεύτερο πείραμα επιλέξαμε σαν σταθερά 4 διεργασίες, αλλάζοντας τον αριθμό των threads. Και στα δύο πειράματα είχαμε ορίσει το πλήθος των συντεταγμένων σε 7 διασκορπισμένο σε όλη την έκταση της εικόνας και βήμα αύξησης μοιρών 0.001. Το αποτέλεσμα των πειραμάτων παρουσιάζεται γραφικά παρακάτω:



ΓΡΑΦΗΜΑ 3: ΠΕΙΡΑΜΑ 1: 5 ΣΤΑΘΕΡΟΣ ΑΡΙΘΜΟΣ THREADS/ ΑΛΛΑΓΗ ΑΡΙΘΜΟΥ ΔΙΕΡΓΑΣΙΩΝ

Όπως βλέπουμε ο συνδυασμός του OpenMp με το MPI φέρει τα πιο αποδοτικά αποτελέσματα, έχοντας τον μικρότερο χρόνο εκτέλεσης σε κάθε περίπτωση. Σημαντικό είναι να εστιάσουμε και στον αριθμό των διεργασιών. Βλέπουμε να επηρεάζει πιο πολύ την υλοποίηση MPI και του συνδυασμού, πράγμα λογικό μιας και το MPI χειρίζεται τον αριθμό των threads, έχοντας έτσι λιγότερες διεργασίες με σταθερά thread υπάρχει περίπτωση να υπάρχουν συνθήκες ανταγωνισμού. Αυτό φαίνεται και όταν ο αριθμός των διεργασιών γίνει 4 από 2 μιας και από το σημείο αυτό και μετά η αλλαγή στον χρόνο εκτέλεσης δεν είναι μεγάλη.



ΓΡΑΦΗΜΑ 4: ΠΕΙΡΑΜΑ 2: 4 ΣΤΑΘΕΡΟΣ ΑΡΙΘΜΟΣ ΔΙΕΡΓΑΣΙΩΝ / ΑΛΛΑΓΗ ΑΡΙΘΜΟΥ THREADS

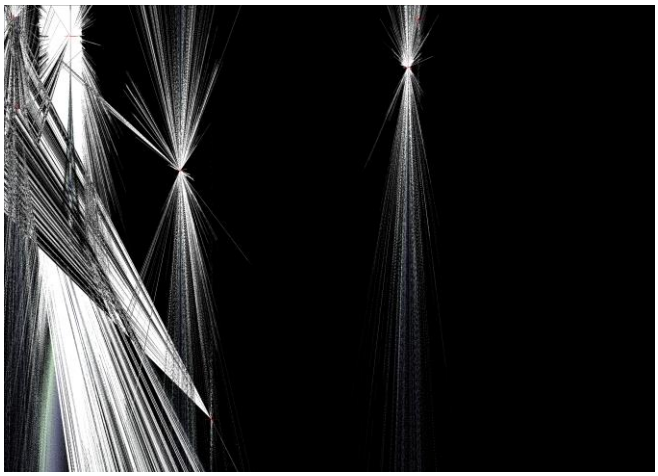
Στο δεύτερο πείραμα τα αποτελέσματα έφεραν και πάλι τον συνδυασμό OpenMp με το MPI στην πρώτη θέση κατέχοντας τον μικρότερο χρόνο εκτέλεσης. Η επιρροή της αλλαγής των thread αυτή την φορά έγινε αρκετά εμφανής στο συνδυασμό των δύο μεθόδων βλέποντας μία αντίστροφη σχέση του χρόνου εκτέλεσης με τον αριθμό των διεργασιών. Δεύτερη θέση κατείχε το OpenMp και τελευταία το MPI όντας όμως και πάλι πολύ υλοποίηση με μέτρο σύγκρισης τον χρόνο εκτέλεσης του σειριακού κώδικα που εκτελέστηκε σε χρόνο 17.7789 δευτερόλεπτα. Τα παραπάνω συμπεράσματα επαληθεύονται και από τον παρακάτω πίνακα που αναγράφει το speed up κάθε υλοποίησης ως προς τον σειριακό κώδικα:

	Speed up					
Πείραμα 1	Αριθμός Διεργασιών	2	4	5	8	15
	MPI	1.02	2.65	2.63	3.02	2.91
	OMP	2.78	2.79	2.82	2.7	2.78
	MPI - Omp	2.25	4.14	4.31	4.15	4.63
Πείραμα 2	Αριθμός thread	2	4	5	8	15
	MPI	2.63	2.66	2.65	2.60	2.63
	OMP	2.77	2.79	2.79	2.76	2.78
	MPI - Omp	3.25	3.78	4.14	4.13	5.17

ΠΙΝΑΚΑΣ 1: SPEED UP ΤΩΝ ΔΥΟ ΠΕΙΡΑΜΑΤΩΝ

Παρατηρούμε πως τα αποτελέσματα του MPI και του OpenMp είναι σχετικά ίδια με διαφορά μιας μονάδας, άλλες φορές υπέρ του MPI, άλλες του OpenMp. Το speed up του συνδυασμού του όμως έχουμε αύξηση δυο μονάδων σε μερικές περιπτώσεις όντας σημαντική βελτίωση. Μοναδική εξαίρεση στη χρήση δύο thread του πειράματος ένα όπου έχουμε μια πτώση της απόδοσης που ίσως να οφείλεται στην έλλειψη πόρων (2 διεργασίες και 5 thread). Παρακάτω

μπορούμε να δούμε και το αποτέλεσμα των πειραμάτων μάς με την τελική χρωματισμένη εικόνα BMP που ήταν ίδια σε κάθε περίπτωση:



ΕΙΚΟΝΑ 1: ΑΠΟΤΕΛΕΣΜΑ ΤΗΣ ΕΚΤΕΛΕΣΗΣ ΤΟΥ ΚΩΔΙΚΑ OPENMP, MPI ΚΑΙ ΣΥΝΔΥΑΣΜΟΥ ΤΟΥΣ ΓΙΑ 7 ΣΗΜΕΙΑ-ΠΑΡΑΤΗΡΗΤΕΣ

IV. ΣΥΜΠΕΡΑΣΜΑΤΑ – ΜΕΛΛΟΝΤΙΚΕΣ ΒΕΛΤΙΩΣΕΙΣ

Το πρόβλημα της ανάλυσης οπτικού πεδίου απαιτεί την επεξεργασία πολλών δεδομένων. Στην δική μας περίπτωση η εικόνα TIFF είχε μέγεθος 3142 επί 2254 και το πλήθος παρατηρητών ήταν σχετικά μικρό, παρ' όλα αυτά έγινε προφανές ότι η παράλληλη επεξεργασία της εικόνας επέφερε σημαντική μείωση στον χρόνο εκτέλεσης. Έτσι σε ακόμη μεγαλύτερες εικόνες και μεγαλύτερες αναλύσεις θα ήταν πολύ επωφελές έως απαραίτητο. Ο κώδικας MPI καθώς και του συνδυασμού OpenMp-MPI είχε την καλύτερη απόδοση όμως, λόγο της χρήσης MPI_Reduce() δεν παρουσιάζει το μέγιστο της απόδοσης, λόγο καθυστέρησης στην επικοινωνία. Μελλοντική βελτίωση θα μπορούσε να είναι η υλοποίηση δικού μας BMP struct με την χρήση MPI_Type_create_struct(). Με αυτό τον τρόπο θα μπορούσε κάθε thread να χρωματίζει απευθείας την εικόνα και να μην χρειάζεται η αποστολή του χρώματος του pixel στο rank 0 σώζοντας έτσι και πολύ μνήμη λόγω της μη δέσμευσης του πίνακα-αντιγράφου της εικόνας.

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

[1] Wheatley David. Cumulative viewshed analysis: a GISbased method for investigating intervisibility, and its archaeological application. s.l. : G Lock and Z Stancic eds GIS and Archaeology: a European Perspective, 1995. σσ. 171 -186.

[2] W. Franklin, C. Ray and S. Mehta. Geometric Algorithms for Siting of Air Defense Missile Batteries. s.l. : A Research Project for Battle, no. 2756, 1994.

[3] Branko Kaucic, Borut Zalik. Comparison of Viewshed Algorithms on Regular Spaced Points. s.l. : Proceedings of the 18th Spring Conference on Computer Graphics, 2002. σσ. 177–183.