Συστήματα Παράλληλης και Κατανεμημένης Επεξεργασίας

Parallelizing Hierarchical Search for Video Motion Estimation using OpenMP and Open MPI

Ευστράτιος Μυλωνάς 534
Ιωάννης Αντωνίου 491

June 18, 2014

Εισαγωγή

Η εργασία μας ξεκίνησε αρχικά με την μελέτη του προγράμματος και την κατανόηση των λειτουργιών του κάθε αρχείου ξεχωριστά και τον τρόπο με τον οποίο αυτά συνδέονται μεταξύ τους για την ορθή λειτουργία του προγράμματος. Αντιμετωπίσαμε πολλές δυσκολίες στο πως λειτουργεί η κάθε συνάρτηση και κατά πόσο χρησιμοποιείτε στο πρόγραμμα μας. Το σειριακό πρόγραμμα είχε χρόνο εκτέλεσης στο μηχάνημα mach-one (όπου εκεί έγιναν και όλες οι παραλληλοποιήσεις για να έχουμε κοινό σημείο αναφοράς) περίπου 6.95 δευτερόλεπτα. Το SNR του σειριακού είναι 40.301233.

Σκοπός της εργασίας μας ήταν να παραλληλοποιήσουμε τον αρχικό σειριακό κώδικα με 4 διαφορετικούς τρόπου παραλληλοποίησης κατά OpenMP και άλλους 4 κατά OpenMPI. Επιλέξαμε να υλοποιήσουμε 4 κατά OpenMP, 2 κατά OpenMPI και 2 με συνδυασμό των 2.

Χρησιμοποιήσαμε διάφορες τεχνικές παραλληλοποίησης ώστε να φτάσουμε στο επιθυμητό αποτέλεσμα, με όσο το δυνατόν λιγότερες αλλαγές στο αρχικό σειριακό πρόγραμμα. Στόχος μας ήταν να μειώσουμε τον χρόνο εκτέλεσης, σε σχέση με το αρχικό πρόγραμμα, κρατώντας το αποτέλεσμα SNR σταθερό.

Εύρεση Κρίσιμου Σημείου με gprof

Αρχικά, η πρώτη μας κίνηση ήταν να βρούμε ποιο σημείο κώδικα επιβαρύνει το πρόγραμμα, ώστε να στοχεύσουμε στην βελτιστοποίηση/παραλληλοποίησή του. Το εργαλείο που μας βοήθησε σε αυτό ήταν το gprof, που μας έδειξε ότι η συνάρτηση hs_motion_estimation2 καταναλώνει περισσότερο χρόνο, με βάση τον συνολικό χρόνο εκτέλεσης, και εκτελείτε περισσότερες φορές, από κάθε άλλη συνάρτηση του προγράμματος, με μεγάλη διαφορά.

Έτσι λοιπόν επικεντρωθήκαμε στο εσωτερικό της συνάρτησης, που βρίσκεται στο αρχείο hs.c, καθώς και στα σημεία που γίνεται η κλήση της συνάρτησης, δηλαδή στην main. Αναλύοντας την συνάρτηση hs_motion_estimation2, παρατηρήσαμε ότι αποτελείται από 3 βασικά κομμάτια κώδικα, που το καθένα περιλαμβάνει πολλαπλές εμφωλευμένες επαναλήψεις for, κάτι που επιβαρύνει την λειτουργία και τον χρόνο εκτέλεσης του προγράμματος. Έτσι προχωράμε στις παρακάτω παραλληλοποιήσεις.

Παραλληλοποιήσεις κατά OpenMP

Πρώτη παραλληλοποίηση

Η πρώτη παραλληλοποίηση έγινε με τη χρήση omp parallel sections , προκειμένου να εκτελέσουμε τα 3 βασικά κομμάτια επαναλήψεων for ταυτόχρονα. Έτσι λοιπόν τοποθετήσαμε στην αρχή της συνάρτησης, κάτω από τις κλήσεις των συναρτήσεων subsampling2 και sumsampling4,την εντολή # pragma omp parallel sections num_threads(3) , ώστε να δημιουργήσουμε 3 omp threads, που το καθένα εκτελεί τα 3 sections που ακολουθούν. Επίσης τοποθετήσαμε την εντολή #pragma omp section, που η καθεμιά περιλαμβάνει τα βασικά κομμάτια επαναλήψεων for.

Παρατηρήσεις: Ο χρόνος εκτέλεσης έχει μειωθεί αισθητά και ο νέος χρόνος εκτέλεσης είναι περίπου 1.88 δευτερόλεπτα. Το SNR όμως έχει κάποια απόκλιση, περίπου κατά -0.7. Έτσι το speedup είναι:
speedup = χρόνος εκτέλεσης σειριακού / χρόνος εκτέλεσης παραλληλοποιημένου
speedup = 6.95 / 1.88 = 3.6968

Δεύτερη παραλληλοποίηση

Η δεύτερη παραλληλοποίηση έγινε με τη χρήση μιας επιπλέον επανάληψης for,τριών βημάτων, που περιλαμβάνει 3 ελέγχους if, ώστε για κάθε τιμή του μετρητή, να εκτελείται ένα από τα βασικά κομμάτια επαναλήψεων for ενώ πάνω από αυτή τοποθετήσαμε #pragma omp parallel for, προκειμένου να εκτελέσουμε τα 3 βασικά κομμάτια επαναλήψεων for ταυτόχρονα.

Παρατηρήσεις: Ο χρόνος εκτέλεσης έχει μειωθεί αισθητά και ο νέος χρόνος εκτέλεσης είναι περίπου 1.89 δευτερόλεπτα. Το SNR όμως έχει κάποια απόκλιση, περίπου κατά -0.1. Έτσι το speedup είναι:
speedup = χρόνος εκτέλεσης σειριακού / χρόνος εκτέλεσης παραλληλοποιημένου
speedup = 6.95 / 1.89 = 3.6960

Τρίτη παραλληλοποίηση

Η τρίτη παραλληλοποίηση έγινε στο αρχείο main.c με την τοποθέτηση της εντολής #pragma omp parallel for private (i) schedule (dynamic, 3)στο σημείο που γίνεται η επανάληψη της κλήσης της συνάρτησης hs_motion_estimation2. Με αυτόν τον τρόπο δημιουργούνται νήματα τα οποία εκτελούν την διαδικασία της επανάληψης με παράλληλο τρόπο.

Παρατηρήσεις: Ο χρόνος εκτέλεσης έχει μειωθεί αισθητά και ο νέος χρόνος εκτέλεσης είναι περίπου 5.05 δευτερόλεπτα. Το SNR όμως έχει κάποια απόκλιση, περίπου κατά +0.5. Έτσι το speedup είναι:
speedup= χρόνος εκτέλεσης σειριακού / χρόνος εκτέλεσης παραλληλοποιημένου
speedup = 6.95 / 5.05 = 1.37623

Τέταρτη παραλληλοποίηση

Η τέταρτη παραλληλοποίηση έγινε στο αρχείοhs.c στην συνάρτηση hs_motion_estimation2 με την τοποθέτηση της εντολής #pragma omp parallel for collapse(2) στις τρείς βασικές επαναλήψεις for που παραλληλοποιεί τις επαναλήψεις for και με το collapse(2) ενώνει

τις 2 for .

Παρατηρήσεις: Ο χρόνος εκτέλεσης έχει μειωθεί αισθητά και ο νέος χρόνος εκτέλεσης είναι περίπου 2.96 δευτερόλεπτα. Το SNR όμως έχει κάποια απόκλιση, περίπου κατά +0.6. Έτσι το speedup είναι:
εν σπεεδυπ = χρόνος εκτέλεσης σειριακού / χρόνος εκτέλεσης παραλληλοποιημένου
speedup = 6.95 / 2.96 = 2.3479

Παραλληλοποιήσεις κατά OpenMPI

Πρώτη παραλληλοποίηση

Η πρώτη παραλληλοποίηση έγινε με τη χρήση 3 MPI Ranks , στην συνάρτηση hs_motion_estimation2.. Έχουμε τοποθετήσει συνθήκες if με τέτοιο τρόπο ώστε το κάθε Rank να περιλαμβάνει και από ένα βασικό κομμάτι επανάληψης for . Οι κλήσεις των MPI_Init και MPI_Finalize για την αρχικοποίηση και τον τερματισμό των ranks, καθώς και ο καθορισμός του αριθμού του κάθε rank και του συνολικού αριθμού των processors γίνεται στη main . Επίσης έχουν τοποθετηθεί κατάλληλοι έλεγχοι για περιπτώσεις όπως πχ. εκτέλεση του προγράμματος με αριθμό επεξεργαστών μικρότερο του 3.

Παρατηρήσεις: Ο χρόνος εκτέλεσης έχει μειωθεί αισθητά και ο νέος χρόνος εκτέλεσης είναι περίπου 2.25 δευτερόλεπτα. Το SNR όμως έχει κάποια απόκλιση, περίπου κατά +0.7. Έτσι το speedup είναι:
speedup = χρόνος εκτέλεσης σειριακού / χρόνος εκτέλεσης παραλληλοποιημένου
speedup = 6.95 / 2.25 = 3.08

Δεύτερη παραλληλοποίηση

Η δεύτερη παραλληλοποίηση έγινε με τη χρήση 2 MPI Ranks , στην συνάρτηση main.. Έχουμε τοποθετήσει συνθήκες if με τέτοιο τρόπο ώστε το κάθε Rank να αναλαμβάνει τις μισές επαναλήψεις από τις συνολικές που γίνονται, στο σημείο που γίνεται και η κλήση της hs_motion_estimation2. Ο διαμοιρασμός γίνεται με ανάλογα με το αν η τιμή του i είναι άρτια ή περιττή. Όπως και πριν τοποθετήθηκαν οι αντίστοιχες συναρτήσεις για την αρχικοποίηση και τον τερματισμό των ranks, την εύρεση των size και rank καθώς και σημεία ελέγχου σωστής κλήσης.

Παρατηρήσεις: Ο χρόνος εκτέλεσης έχει μειωθεί αισθητά και ο νέος χρόνος εκτέλεσης είναι περίπου 1,75 δευτερόλεπτα. Το SNR όμως έχει κάποια απόκλιση, περίπου κατά +0.7. Έτσι το speedup είναι:
speedup = χρόνος εκτέλεσης σειριακού / χρόνος εκτέλεσης παραλληλοποιημένου
speedup = 6.95 / 1.75 = 3.97

Παραλληλοποιήσεις κατά OpenMP + OpenMPI

Πρώτη παραλληλοποίηση

Η πρώτη μικτή παραλληλοποίηση έγινε με τη χρήση 2 MPI Ranks , στην συνάρτηση main.σε συνδυασμό με την χρήση της μεθόδου με την επιπλέον for που χρησιμοποιήθηκε στη δεύτερη παραλληλοποίηση με OpenMP . Στην main , έχουμε τον διαμοιρασμό των επαναλήψεων της for που καλεί την συνάρτηση

hs_motion_estimation2, όπως είχε γίνει στην δεύτερη παραλληλοποίηση κατά OpenMPI .

Παρατηρήσεις: Ο χρόνος εκτέλεσης έχει μειωθεί αισθητά και ο νέος χρόνος εκτέλεσης είναι περίπου 1,89 δευτερόλεπτα. Το SNR όμως έχει κάποια απόκλιση, περίπου κατά +0.1. Έτσι το speedup είναι:
speedup = χρόνος εκτέλεσης σειριακού / χρόνος εκτέλεσης παραλληλοποιημένου
speedup = 6.95 / 1.89 = 3,68

Δεύτερη παραλληλοποίηση

Η δεύτερη παραλληλοποίηση έγινε με τη χρήση 3 MPI Ranks , στην συνάρτηση hs_motion_estimation2 σε συνδυασμό με την χρήση της εντολής #pragma omp parallel for private (i) schedule (dynamic, 3) στο σημείο που γίνεται η επανάληψη της κλήσης της συνάρτησης hs_motion_estimation2 . Έχουμε τοποθετήσει συνθήκες if με τέτοιο τρόπο ώστε το κάθε Rank να περιλαμβάνει και από ένα βασικό κομμάτι επανάληψης for , όπως και στην πρώτη παραλληλοποίηση κατά OpenMPI.

Παρατηρήσεις: Ο χρόνος εκτέλεσης έχει μειωθεί αισθητά και ο νέος χρόνος εκτέλεσης είναι περίπου 2.88 δευτερόλεπτα. Το SNR όμως έχει κάποια απόκλιση, περίπου κατά +0.5. Έτσι το speedup είναι:
speedup = χρόνος εκτέλεσης σειριακού / χρόνος εκτέλεσης παραλληλοποιημένου
speedup = 6.95 / 2.88 = 2.413

Πρόσθετες αλλαγές στις παραλληλοποιήσεις

Σε κάθε παραλληλοποίηση έγιναν οι κατάλληλες αλλαγές και προσθήκες στα Makefile , προκειμένου με τις εντολές make, make clean και make execute να εκτελούνται κανονικά όπως θα έπρεπε. Προσθέσαμε πρόσθετες ετικέτες /flags για την σωστή μεταγλώττιση και σύνδεση του προγράμματος, καθώς και αλλαγή μεταγλωττιστή, όπου αυτό ήταν απαραίτητο πχ. αλλαγή του gcc σε mpicc .

Σχολιασμός Αποτελεσμάτων

Αν και καταφέραμε να μειώσουμε αρκετά το χρόνο εκτέλεσης των προγραμμάτων, χρησιμοποιώντας παραλληλοποιήσεις OpenMP, OpenMPI καθώς και συνδυαστικές μεθόδους των δύο, δυσκολευτήκαμε στο να κρατήσουμε την τιμή του αποτελέσματος SNR σταθερή. Μπορέσαμε όμως και στις οκτώ παραλληλοποιήσεις, που αναλύθηκαν παραπάνω, να το διατηρήσουμε την απόκληση κατά 0.8 μονάδες, σε απόλυτη τιμή, ώστε να μην ξεπερνάει το αρχικό SNR πάνω από μια μονάδα.

Θεωρούμε ότι οι παραλληλοποιήσιες πραγματοποιήθηκαν σε ικανοποιητικό επίπεδο, με βάση τον περιορισμένο χρόνο που είχαμε και το φόρτο εργασίας. Καλύψαμε κατά μεγάλο ποσοστό τους βασικούς σκοπούς της εργασίας που μας ανατέθηκε.

Ακόμα όλες οι παραλληλοποιήσεις μεταγλωττίζονται επιτυχώς χωρίς κάποιο warning/error κατά την μεταγλώττιση.

Δυσκολίες που αντιμετωπίσαμε

Στην πορεία υλοποίησης της εργασίας αντιμετωπίσαμε διάφορα προβλήματα, κυρίως με το συγχρονισμό των νημάτων/ranks , κάτι που το παρατηρούσαμε είτε από την υπέρμετρη απόκληση του αποτελέσματος, είτε από τον μεγάλο χρόνο εκτέλεσης του προγράμματος. Επίσης είχαμε δυσκολία ώστε να κατανοήσουμε τον κώδικα και πως αυτός λειτουργεί.

Ο χρόνος υλοποίησης της εργασίας αυτής ήταν μια εβδομάδα.

Επίλογος

Μετά από την ενασχόληση με το αντικείμενο της εργασίας μας, μπορούμε να εξάγουμε το συμπέρασμα ότι η παραλληλοποίηση ενός σύνθετου προγράμματος είναι πολύ σημαντική, δεδομένου ότι επιταχύνει πολύ τον χρόνο εκτέλεσης του προγράμματος και το κάνει να χρησιμοποιεί πιο αποδοτικά τους διαθέσιμους πόρους του συστήματος. Οι τρόποι με τους οποίους μπορεί να χρησιμοποιηθεί η παραλληλοποίηση είναι πάρα πολλοί, καθώς και οι διαφορετικοί διαθέσιμοι μέθοδοι και εργαλεία που υπάρχουν. Μάθαμε να χειριζόμαστε καλύτερα τις μεθόδους παραλληλοποίησης και να τις εφαρμόζουμε στα σωστά σημεία ενός κώδικα.

Μαζί με τους κώδικες (χωρισμένους σε φακέλους), παραθέτοντας και print screens με ενδεικτικές εκτελέσεις και αποτελέσματα για καθεμιά.