Προγραμματισμός Συστημάτων Υψηλών Επιδόσεων (ECE 415)
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

Διδάσκων: Χρήστος Δ. Αντωνόπουλος

2η Εργαστηριακή Άσκηση – Προθεσμία 1/11/2022, 23:59

Στόχος: Παραλληλοποίηση ολοκληρωμένης ακολουθιακής εφαρμογής με χρήση OpenMP. Πειραματική αξιολόγηση σε πολυπύρηνο επεξεργαστή.

Εισαγωγή:
Όπως συζητήσαμε στο μάθημα, ένα βασικό πλεονέκτημα του μοντέλου προγραμματισμού OpenMP είναι ότι επιτρέπει την εύκολη, σταδιακή παραλληλοποίηση ακολουθιακών εφαρμογών. Στο συγκεκριμένο εργαστήριο σας δίνεται ο ακολουθιακός κώδικας μιας υλοποίησης του αλγόριθμο clustering k-means. Ο αλγόριθμος ομαδοποιεί n παρατηρήσεις σε k clusters. Κάθε cluster χαρακτηρίζεται από τις συντεταγμένες του "κέντρου" (centroid) του. Καθεμιά από τις n παρατηρήσεις ανατίθεται στο cluster από το centroid του οποίου έχει τη μικρότερη απόσταση. Επίσης σας δίνονται πολλαπλά διαφορετικά αρχεία εισόδου για διαφορετικά μεγέθη προβλημάτων (από απλά έως πιο ρεαλιστικά).

Η εφαρμογή είναι δεκτική σε παραλληλοποίηση. Ζητείται να παραλληλοποιήσετε την εφαρμογή με OpenMP και να αξιολογήσετε την επίδοσή της σε πολυπύρηνο σύστημα με επεξεργαστή Intel Xeon E5-2695. Όλη η ανάπτυξη θα γίνει στα συστήματα inf-mars1 (10.64.82.31) και inf-mars2 (10.64.82.32) ενώ οι τελικές μετρήσεις στο csl-artemis (10.64.82.65). Σε κάθε περίπτωση, το τελικό εκτελέσιμο θα πρέπει να παράγεται με τον Intel C compiler. Χρησιμοποιήστε σε όλα τα πειράματα τις επιλογές βελτιστοποιήσεων του μεταγλωττιστή που δίνουν τα καλύτερα αποτελέσματα (σε χρόνο εκτέλεσης) για τον αρχικό ακολουθιακό κώδικα. Κρατήστε ένα αντίγραφο του αρχικού ακολουθιακού κώδικα ως σημείο αναφοράς (για συγκρίσεις ορθότητας και επίδοσης).

Για τον τρόπο που εκτελείται η εφαρμογή δείτε το σχετικό README.

Μπορείτε να αλλάξετε τον αριθμό νημάτων που χρησιμοποιεί το εκτελέσιμο OpenMP θέτοντας κατάλληλα τη μεταβλητή περιβάλλοντος OMP_NUM_THREADS. Π.χ.:

```
export OMP_NUM_THREADS=4
```

Κάθε φορά που μετράτε επίδοση βεβαιωθείτε ότι δεν τρέχει κάποιος άλλος ταυτόχρονα.

Ζητούμενα:
Το πρώτο λογικό βήμα είναι να εκτελέσετε profiling και να εντοπίσετε τα σημεία (loops) στα οποία αξίζει να επικεντρώσετε την προσοχή σας. Προχωρήστε σταδιακά, ξεκινώντας από τα loops στα οποία αφιερώνεται πολύς χρόνος εκτέλεσης και προχωρήστε σταδικά σε αυτά με λιγότερο χρόνο εκτέλεσης.

Για κάθε loop που εντοπίζετε, ελέγξτε αν είναι παραλληλοποιήσιμο. Αν είναι, προσθέστε τα κατάλληλα OpenMP directives. Προσέξτε ιδιαίτερα τυχόν μεταβλητές που πρέπει να δηλωθούν private. Πειραματιστείτε με την κατάλληλη πολιτική χρονοδρομολόγησης και το chunk για κάθε loop. Αφού βεβαιωθείτε ότι ο παράλληλος κώδικας δίνει σωστά αποτελέσματα (συγκρίνοντας με τον ακολουθιακό), εξετάστε αν συμφέρει (από την άποψη της επίδοσης) η παραλληλοποίηση ή αν τυχόν

οδηγεί σε μεγαλύτερο χρόνο εκτέλεσης. Δοκιμάστε κάθε φορά να εκτελέσετε με 1, 4, 8, 16, 32, και 64 threads στο csl-artemis (1, 2, 4, 8 threads στα inf-mars1 και inf-mars2). Μόνο αφού τελειώσετε με την παραλληλοποίηση και την εκτίμηση επίδοσης ενός loop είναι σκόπιμο να προχωρήσετε σε επόμενο.

Αφού παραλληλοποιήσετε όλα τα loops των οποίων η παράλληλη εκτέλεση οδηγεί σε καλύτερη επίδοση, ασχοληθείτε με τυχόν "συνολικές βελτιστοποιήσεις" που μπορούν να γίνουν (για παράδειγμα – και όχι μόνο – με το εύρος των parallel regions, τυχόν implicit barriers που θα μπορούσαν να αφαιρεθούν κλπ). Επίσης σκεφτείτε τυχόν αλλαγές στις δομές δεδομένων και την αρχική υλοποίηση που θα μπορούσαν να προσφέρουν παραπάνω παραλληλισμό, καλύτερη επίδοση κλπ.

Στα πλαίσια της πειραματικής αξιολόγησης, δώστε το χρόνο εκτέλεσης του αρχικού ακολουθιακού προγράμματος, καθώς και το χρόνο εκτέλεσης του OpenMP προγράμματος με 1, 4, 8, 16, 32, 64 threads. Επαναλάβετε κάθε πείραμα πολλαπλές φορές και δώστε μέση τιμή και τυπική απόκλιση των μετρήσεων. Προσπαθήστε να σχολιάσετε τα αποτελέσματα. Σημειώστε ότι:

α) Ο χρόνος εκτέλεσης του OpenMP προγράμματος με 1 thread ενδέχεται να διαφέρει από αυτόν του ακολουθιακού προγράμματος λόγω του overhead της παραλληλοποίησης.

β) Στους χρόνους ΔΕ θα πρέπει να συμπεριλάβετε το I/O (είσοδο / έξοδο από / σε αρχεία).

γ) Ο επεξεργατής των inf-mars (i7-4820K) είναι είναι 4-way multicore. Συνεπώς όταν εκτελείτε από 1 έως 4 threads το λειτουργικό φροντίζει να αναθέσει καθένα σε διαφορετικό core. Το csl-artemis, αντίστοιχα, έχει 2 επεξεργαστές, έκαστος 16-way multicore.

δ) Και στα 2 μηχανήματα κάθε core είναι 2-way SMT (ή hyperthreaded σύμφωνα με την ορολογία της Intel). Συνεπώς, όταν εκτελείτε την εφαρμογή με περισσότερα threads από cores, κάποια μοιράζονται το ίδιο core.

Παράδοση:

Πρέπει να παραδώσετε:

- Τον τελικό κώδικα.
- Αναφορά στην οποία θα αναλύετε τη στρατηγική παραλληλοποίησης που ακολουθήσατε και τα σημεία που παραλληλοποιήσατε. Μην παραλείψετε να αναφέρετε και τυχόν βελτιστοποιήσεις / παραλληλοποιήσεις που δοκιμάσατε χωρίς καλά αποτελέσματα στο χρόνο εκτέλεσης. Στην αναφορά συμπεριλάβετε και την πειραματική αξιολόγηση. Επίσης, αναφέρετε τα flags του μεταγλωττιστή που χρησιμοποιήσατε.

Δημιουργήστε ένα αρχείο .tar.gz με τα παραπάνω περιεχόμενα και όνομα <όνομα1>_<AEM1>_<όνομα2>_<AEM2>_lab2.tar.gz. Ανεβάστε το εμπρόθεσμα στο e-class.