ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Παράλληλες Αρχιτεκτονικές Υπολογισμού για Μηχανική Μάθηση

Ε.ΔΕ.Μ²

Ακαδημαϊκό Έτος 2019-20

http://www.cslab.ece.ntua.gr/courses/parml

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2
Παράλληλη Επεξεργασία σε Αρχιτεκτονικές Κοινής Μνήμης με OpenMP

1 Εισαγωγικά

Στα πλαίσια αυτής της εργασίας θα εξοικειωθείτε με τον παράλληλο προγραμματισμό σε αρχιτεκτονικές κοινής μνήμης με το προγραμματιστικό μοντέλο OpenMP. Ο βοηθητικός κώδικας της εργασίας βρίσκεται στον scirouter στον κατάλογο `/home/parml/shared/lab2_omp`. Ο κατάλογος περιέχει 4 υποκαταλόγους, έναν για κάθε άσκηση: `ex1`, `ex2`, `ex3` και `ex4`. Για κάθε άσκηση, θα βρείτε και τα αντίστοιχα Torque scripts για τη μεταγλώττισή και εκτέλεσή τους. Η μεταγλώττιση και εκτέλεση θα γίνει στα μηχανήματα `clones`.

Σαν πρώτο βήμα, αντιγράψτε το φάκελο `/home/parml/shared/lab2_omp` στο home directory σας στον `scirouter`.

2 Ζητούμενα

2.1 "Hello World!"

Στο OpenMP, μπορούμε να ορίσουμε το πλήθος των νημάτων μιας παράλληλης περιοχής με τρεις τρόπους:

- Θέτοντας την κατάλληλη τιμή στη μεταβλητή περιβάλλοντος `OMP_NUM_THREADS`
- Με κλήση της συνάρτησης βιβλιοθήκης `omp_set_num_threads (int num_threads)` στον κώδικα
- Με τη χρήση του clause (όρου) `num_threads()` στην οδηγία `#pragma_omp_parallel` στον κώδικα

Επιπλέον, μπορούμε να μάθουμε το αναγνωριστικό (ID) ενός νήματος με μια κλήση της συνάρτησης βιβλιοθήκης `int omp_get_thread_num()`.
Στον κατάλογο `ex1` σας δίνεται το αρχείο `hello_world.c`. Στόχος της άσκησης είναι:

1. να ορίσετε μια παράλληλη περιοχή με OpenMP

2. να βρείτε το πλήθος των νημάτων στην παράλληλ περιοχή
3. να βρείτε το ID του κάθε νήματος
4. να μεταγλωττίσετε τον κώδικα

```
$ gcc -O3 -fopemp HelloWorld.c -o helloworld
```

5. και να τρέξετε το παράλληλο πρόγραμμα με 8 νήματα

```
$ export OMP_NUM_THREADS=8
$ ./helloworld
```

Ερώτηση: Τι τιμή επιστρέφει η κλήση στη συνάρτηση `omp_get_num_threads()` μέσα σε μία παράλληλη περιοχή; Τι τιμή επιστρέφει έξω από μία παράλληλη περιοχή;

2.2 Επισημείωση μεταβλητών

Στο OpenMP, ο προγραμματιστής μπορεί να δηλώσει αν μια μεταβλητή σε μία παράλληλη περιοχή θα είναι `shared`, `private`, `firstprivate` κ.ά. Η επισημείωση των μεταβλητών γίνεται ως clause στις οδηγίες του OpenMP (`#pragma omp ...`) προς τον μεταγλωττιστή.

- Μια `shared` μεταβλητή είναι κοινή για όλα τα νήματα - διαβάζουν και γράφουν στην ίδια θέση μνήμης
- Μια `private` μεταβλητή είναι ιδιωτική για κάθε νήμα - κάθε νήμα διαβάζει και γράφει σε άλλη θέση μνήμης
- Μια `firstprivate` μεταβλητή είναι ιδιωτική, αλλά έχει αρχικοποιηθεί με την τιμή της shared μεταβλητής

Στον κατάλογο `ex2` σας δίνεται το αρχείο `Variables.c`. Στόχος της άσκησης είναι:

1. Να μεταγλωττίσετε και να εκτελέσετε το δοσμένο παράλληλο πρόγραμμα με μία shared μεταβλητή και να δείτε τα αποτελέσματα της εκτέλεσης.
2. Να τροποποιήσετε τη shared μεταβλητή αρχικά i) σε private και στη συνέχεια ii) σε firstprivate και να δείτε τα αποτελέσματα της εκτέλεσης.

   (Προσοχή: κάθε φορά που αλλάζετε τον κώδικά σας, θα χρειαστεί να μεταγλωττίστε ξανά και να παράξετε καινούργιο εκτελέσιμο.)

Ερώτηση: Εξηγείστε πώς προκύπτει η τιμή της μεταβλητής `x` εντός και εκτός της παράλληλης περιοχής, όταν η μεταβλητή είναι `shared`, `private` και `firstprivate` αντίστοιχα.

2.3 Άθροισμα διανυσμάτων

Οι δομές `for` είναι πολύ συνηθισμένες στον προγραμματισμό και ειδικά στον επιστημονικό προγραμματισμό. Η παραλληλοποίηση βρόχων στο OpenMP είναι εύκολη για τον προγραμματιστή, αφού αρκεί η προσθήκη της οδηγίας `#pragma omp for` πριν την εντολή `for`.
Στον κατάλογο `ex3` σας δίνεται σειριακή υλοποίηση για πρόσθεση διανυσμάτων, στο αρχείο `VecAdd.c`. Στόχος της άσκησης είναι:

1. Να παραλληλοποιήσετε το σειριακό κώδικα για την πρόσθεση διανσυμάτων με τη χρήση του OpenMP
2. Να εκτελέσετε τον παράλληλο κώδικα με 1, 2, 4, 8 και να παρατηρήσετε την παράλληλη επίδοση

Ερώτηση 1: Υπάρχει βελτίωση της επίδοσης με χρήση περισσότερων νημάτων;
Ερώτηση 2: Αυξήστε το μέγεθος των διανυσμάτων και επαναλάβετε τη μεταγλώττιση και την εκτέλεση. Τι παρατηρείτε ως προς την επίδοση;

2.4 Εσωτερικό γινόμενο διανυσμάτων

Στο παράδειγμα της πρόσθεσης διανυσμάτων, τα νήματα κάνουν υπολογισμούς ανεξάρτητα και δεν απαιτείται συνδυασμός των αποτελεσμάτων. Αυτό δε συμβαίνει πάντα.
Στον κατάλογο `ex4` σας δίνεται σειριακή υλοποίηση για πρόσθεση διανυσμάτων, στο αρχείο `DotProduct.c`. Στόχος της άσκησης είναι:

1. Να παραλληλοποιήσετε το σειριακό κώδικα για το εσωτερικό γινόμενο διανυσμάτων με τη χρήση του OpenMP
2. Να εκτελέσετε τον παράλληλο κώδικα με 1, 2, 4, 8 και να παρατηρήσετε την παράλληλη επίδοση

Ερώτηση 1: Για την επίλυση αυτού του προβλήματος, υπάρχουν περισσότερες από μία δυνατές υλοποιήσεις στο OpenMP. Μπορείτε να τις περιγράψετε;
Ερώτηση 2: Τι παρατηρείτε ως προς την παράλληλη επίδοση; Υπάρχει διαφορά στην επιτάχυνση σε σχέση με την περίπτωση του αθροίσματος διανυσμάτων; Αν ναι, από πού προκύπτει και γιατί;

3 Υποδείξεις και διευκρινίσεις

Θα τρέξετε τον κώδικά σας στα μηχανήματα του εργαστηρίου `clones` της ουράς `parlab`. Για τις οδηγίες χρήσης, ανατρέξτε στον Οδηγό Εργαστηρίου.