ECE445 Παράλληλοι και Δικτυακοί Υπολογισμοί
Χειμερινό Εξάμηνο 2022-2023
Εργασία 2

Εισαγωγικά

(Ημερομηνία Παράδοσης: Τετάρτη 12.01.2023, 12:00μμ)
Ακολουθείστε το πρότυπο κείμενο (template.docx) για να γράψετε τις απαντήσεις σας στις παρακάτω ερωτήσεις και την αναφορά σας στα πειραματικά αποτελέσματα των προγραμμάτων σας.
Γράψτε καθαρά τα ονοματεπώνυμά σας και τα ΑΕΜ σας στην πρώτη σελίδα. Το κείμενο σας πρέπει να είναι καλογραμμένο και ευανάγνωστο ενώ θα πρέπει να δικαιολογείτε **ΠΛΗΡΩΣ** τα βήματα που ακολουθήσατε, και να σχολιάζετε τα αποτελέσματα από κάθε άσκηση (πίνακες, γραφικές παραστάσεις κλπ.
Προτείνεται να προγραμματίστε σε C/C++ και σε περιβάλλον Linux, χρησιμοποιείστε Makefile για compile/link/execution των προγραμμάτων σας.
Ετοιμάστε την αναφορά (report) σας, γράφοντας τις θεωρητικές λύσεις των προβλημάτων, περιγράφοντας την υλοποίηση και το μηχάνημα που τρέξατε τα προγράμματά σας και κάνατε τα πειράματά σας.
Καταγράψτε τις παρατηρήσεις σας, και παρουσιάστε τα αποτελέσματα σε γραφήματα και πίνακες ώστε να φαίνεται η κλιμάκωση του χρόνου σε σχέση με το μέγεθος των εισόδων.
Παραδώστε μια φορά την εργασία σας μέσω eclass σε ένα zip αρχείο (ergasia1_AEM1_AEM2.zip), το οποίο θα περιέχει το κείμενο με τις λύσεις/αποτελέσματα και σχόλια σας, τον κώδικά σας (.c, .h αρχεία και το Makefile σας). Σημειώνεται ότι οι ομάδες δεν αλλάζουν στη διάρκεια του εξαμήνου.

Άσκηση 1 – Εισαγωγή στον FFT (θεωρητική) (10 μονάδες)

a) Γράψτε και μελετήστε τον σειριακό αλγόριθμο για FFT για $n(=2^K)$ στοιχεία. Υπολογίστε το κόστος του.

b) Γράψτε και μελετήστε τον παράλληλο αλγόριθμο για n στοιχεία σε $p(=2^m)$ παράλληλες εργασίες. Υπολογίστε το κόστος του σε συνάρτηση των n και p.

c) Υπολογίστε την χρονοβελτίωση και την απόδοση του αλγόριθμου.

d) Υπολογίστε το ζευγάρι n,p που βελτιστοποιεί την χρονοβελτίωση (σε συνάρτηση των παραμέτρων t*).

e) Υπολογίστε το ζευγάρι n,p που βελτιστοποιεί την απόδοση (σε συνάρτηση των παραμέτρων t*).

Άσκηση 2 – Σειριακή Υλοποίηση (20 μονάδες)

Θεωρούμε το πρόβλημα του ταχύ μετασχηματισμού Fourier $n(=2^k)$ στοιχείων. Υλοποιείστε την σειριακή έκδοση του αλγορίθμου. Εκτελέστε το πρόγραμμα σας για k=10,(1),20 και επιβεβαιώστε την πολυπλοκότητα του αλγόριθμου που μελετήσατε στο σετ ασκήσεων 2.

Άσκηση 3 – Παράλληλη Υλοποίηση (20 μονάδες)

Υλοποιήστε τον παράλληλο αλγόριθμο του FFT για $n(=2^k)$ στοιχεία σε $p(=2^m)$ νήματα με k όπως στην άσκηση 1 (παραπάνω) και m=1, 2, 3, 4, 5, 6... Προχωρήστε σε αριθμό νημάτων μέχρι να αρχίσετε να έχετε ορατό overhead.

Εγκαταστήστε το OpenMP και Intel Parallel Studio για να μπορέσετε να τρέξετε το πρόγραμμα σας παράλληλα και να μελετήσετε με το VTunes τη συμπεριφορά του.
Από την ιστοσελίδα της INTEL:
https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler.html#gs.kojbqg

Άσκηση 4 – Πειραματική Αξιολόγηση (20 μονάδες)

Φτιάξτε πίνακες όπως τον Πίνακα 1 και σημειώστε τους χρόνους για σειριακή και παράλληλη εκτέλεση και μέσα σε παρένθεση το speedup που είχατε σε κάθε περίπτωση. Κάντε γραφική παράσταση για το «πλήθος των threads vs καλύτερο speedup». Τι παρατηρείτε; Συμφωνούν τα αποτελέσματά σας με τη θεωρητική ανάλυση του σετ ασκήσεων 2;

k vs m	0	1	2	3	4	…
10	105	60 (1,75)				
11						
…						

Για την περίπτωση που είχατε το μεγαλύτερο speedup, συγκρίνετε την υλοποίηση σας με την βιβλιοθήκη FFTW (https://www.fftw.org). Δείτε το manual (https://www.fftw.org/fftw3.pdf) στο κεφάλαιο 2 για την καλέσετε και να μετρήσετε το χρόνο. Δεν χρειάζεται να τρέξετε την παράλληλη υλοποίηση της FFTW αλλά την σειριακή σε σύγκριση με την δική σας παράλληλη. Τέλος η FFTW έχει και εκδόσεις και για Windows και Linux.

Άσκηση 5 – Παράλληλη Αριθμητική Ολοκλήρωση (30 μονάδες)

Ο σειριακός κώδικας στο mc_serial.c υπολογίζει το ολοκλήρωμα μιας συνάρτησης χρησιμοποιώντας τη μέθοδο Monte Carlo. Εφαρμόστε στο mc_parallel.c μια παράλληλη έκδοση του κώδικα χρησιμοποιώντας το OpenMP. Εξετάστε την κλιμάκωση του κώδικά σας. Συμβουλές:

- Αντικαταστήστε την drand48 με τη αντίστοιχη ασφαλούς λειτουργίας για νήματα erand48.
- Χρησιμοποιήστε ένα ιδιωτικό buffer για την erand48 σε κάθε νήμα του OpenMP.
- Χρησιμοποιήστε διαφορετικό seed για να αρχικοποιήσετε κάθε buffer για το erand48.

```
1  // prototype
2  double erand48(unsigned short xsubi[3]);
3
4  // declaration and initialization of buffer
5  unsigned short buffer[3];
6  buffer[0] = 0;
7  buffer[1] = 0;
8  buffer[2] = <thread related number>;
9
10 // usage
11 xi = erand48(buffer);
```