ECE445: ΠΑΡΑΛΛΗΛΟΙ ΚΑΙ ΔΙΚΤΥΑΚΟΙ ΥΠΟΛΟΓΙΣΜΟΙ
Χειμερινό Εξάμηνο 2024-2025

Εργασία 2:

Προγραμματισμός με OpenMP

(*Ημερομηνία Παράδοσης: Κυριακή 08.12.2024, 23:55)*

Γενικές Οδηγίες για όλες τις εργασίες:

Ακολουθείστε το υπόδειγμα κειμένου (template.docx) καθώς και αυτό αποτελεί μέρος στο βαθμό σας.

Εισάγετε Captions (References → Insert Caption) σε εικόνες και πίνακες και όταν να αναφέρετε μέσα στο κείμενο κάντε χρήση του References → Crooss-Reference, ώστε να συνδέεται σωστά το κείμενο με τα εν λόγω αντικείμενα.

Γράψτε καθαρά τα ονοματεπώνυμά σας και τα ΑΕΜ σας στην πρώτη σελίδα.

Στις θεωρητικές ασκήσεις γράψτε αναλυτικά τις απαντήσεις σας με δικά σας λόγια και αναφέρετε τις πηγές από όπου τις αντλήσατε.

Στις προγραμματιστικές, εξηγείστε τον αλγόριθμο που προγραμματίσατε και παρουσιάστε αναλυτικά τα πειραματικά αποτελέσματα των προγραμμάτων σας.

Σε περίπτωση που πήρατε έτοιμο κώδικα, να αναφέρετε τη συγκεκριμένη πηγή (π.χ., github κλπ.) στη βιβλιογραφία, ωστόσο να εξηγήσετε το αλγόριθμο που υλοποιεί.

Περιγράψτε την υλοποίησή σας αλλά και το μηχάνημα στο οποίο δουλέψατε (χαρακτηριστικά υλικού, OS, compiler κλπ.).

Προγραμματίστε σε C και σε περιβάλλον Linux, χρησιμοποιείστε αρχεία Makefile για compile/link/execution των προγραμμάτων σας.

Καταγράψτε τις παρατηρήσεις σας, και παρουσιάστε τα αποτελέσματα με πίνακες, γραφικές παραστάσεις.

Γραφήματα και γραφικές παραστάσεις πρέπει να έχουν δημιουργηθεί με λογισμικό και να μην είναι σκαναρισμένα.

Γενικά 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. ***(20 μον)***

Παραδώστε την εργασία σας μέσω eclass σε ένα zip αρχείο (hw1_AEM1_AEM2_AEM3.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. (30)

Γράψτε ένα πρόγραμμα σε C (ask1.c) που υλοποιεί παράλληλο προγραμματισμό με OpenMP και καλώντας τις κατάλληλες συναρτήσεις θα εμφανίζει στην οθόνη με τον ίδιο τρόπο όπως στην Εικόνα 1:

a) (2) OpenMP version που υπάρχει στον υπολογιστή που τρέχει το πρόγραμμα

b) (2) πόσους επεξεργαστές έχει ο υπολογιστής που εκτελεί το πρόγραμμα

c) (2) πλήθος νημάτων που συμμετέχουν στην εκτέλεση

d) (2) μέγιστο πλήθος νημάτων του υπολογιστή

e) (2) επιβεβαίωση ότι τα μηνύματα εμφανίζονται από παράλληλη περιοχή

f) (2) αν υπάρχει δυνατότητα για dynamic threading

g) (2) μέγιστο πλήθος επιπέδων εμφωλευμένου παραλληλισμού

h) (2) μήνυμα από το master thread : «HELLO. I am the Master thread. I created all participating threads. »

i) (4) μήνυμα από όλα τα νήματα : «I am thread * and worked for ** msec.» , όπου * = το id τους και ** ο χρόνος εκτέλεσης τους.
j) (10) Μέσα στην αναφορά σας γράψτε ποιες συναρτήσεις χρησιμοποιήσατε και προσθέστε ένα πίνακα, στον οποίον θα φαίνεται η παραπάνω πληροφορία και ο αντίστοιχος χρόνος εκτέλεσης στον υπολογιστή σας.
Υπόδειξη: Χρησιμοποιήστε σαν βάση το «Test Your OpenMP» που βρίσκεται στα «Έγγραφα→OpenMP material». Αποφύγετε τους ελέγχους με βάση το id και χρησιμοποιήστε τις κατάλληλες συναρτήσεις για να διαφοροποιήσετε την εκτέλεση ανά νήμα.

```
HELLO! I am the Master thread. I created all participating threads.
I am thread 5 and worked for 0.000000 msec.
OpenMP version: 201107. Information provided by thread 8 .
Number of processors = 12
Number of threads = 12
Max threads = 12
In parallel? = 1
Dynamic threads enabled? = 0
Nested parallelism levels supported = 1
I am thread 1 and worked for 0.000000 msec.
I am thread 4 and worked for 0.000000 msec.
I am thread 0 and worked for 0.030994 msec.
I am thread 11 and worked for 0.005007 msec.
I am thread 3 and worked for 0.000954 msec.
I am thread 2 and worked for 0.000954 msec.
I am thread 8 and worked for 0.124931 msec.
I am thread 10 and worked for 0.000954 msec.
I am thread 7 and worked for 0.000000 msec.
I am thread 9 and worked for 0.000000 msec.
I am thread 6 and worked for 0.000000 msec.
```

Εικόνα 1. Αποτελέσματα από την εκτέλεση του ask1.

Άσκηση 2. (260)
Γράψτε ένα πρόγραμμα σε C (ask2_s.c) **(30)** που υπολογίζει σειριακά το γινόμενο δύο πινάκων A και B με διαστάσεις MxK και KxN αντίστοιχα, όπου A(i,j) = i+j και B(i,j)=i*j και C=A*B.
Έπειτα μετατρέψτε το σε παράλληλο με OpenMP (ask2_p.c).
Το πρόγραμμά σας θα πρέπει να ικανοποιεί τα παρακάτω:
a) (20) η αρχικοποίηση των τριών πινάκων να πραγματοποιείται ταυτόχρονα (όσο το επιτρέπει το πλήθος των νημάτων που δημιουργούνται στο πρόγραμμα) αλλά μόνο από ένα 1 νήμα για κάθε πίνακα (δηλ. κάθε νήμα θα τρέχει σειριακό κώδικα στο συγκεκριμένο κομμάτι του προγράμματος). Θα πρέπει να εμφανίζεται σχετικό μήνυμα, π.χ., «Το νήμα 1 αρχικοποιεί τον πίνακα Α», «Το νήμα 0 αρχικοποιεί τον πίνακα Β» κλπ.
b) (5) αμέσως μετά την αρχικοποίηση των πινάκων, για κάθε νήμα θα εμφανίζεται το μήνυμα «Thread … is starting computations.»
c) (20) θα μετρά το wall clock time της εκτέλεσης του προγράμματος (παράλληλο χρόνο).
d) (20) θα μετρά το wall clock time του κάθε νήματος και θα υπολογίζει το συνολικό παράλληλο χρόνο.
e) (10) θα μετρά το πλήθος των πράξεων που κάνει κάθε νήμα.
f) (10) θα δέχεται ως ορίσματα το πλήθος των threads και το μέγεθος του chunk (πλήθος επαναλήψεων που αναλαμβάνει κάθε νήμα σε επαναληπτική διαδικασία for), οπότε θα το καλείτε στο command line: ./ask2_p 2 10, όπου nthreads=2, chunk=10
g) (5) το master thread θα εμφανίσει (εκτός παράλληλης διαδικασίας) στην οθόνη τα μηνύματα:
“Starting matrix-matrix multiplication example with …. threads, M= …, K= …, N= …, chunk= …”
“Computations are done.”
****
Time per thread:
Time for thread 0 = …… sec. Number of FLOp = ….
Time for thread 1 = …… sec. Number of FLOp = ….
Total Parallel Time = …… sec.
Wall Clock Time = …… sec.
****

Μελέτη συμπεριφοράς του προγράμματός σας και κατανόηση των σχετικών OpenMP συναρτήσεων:
Α) (80) Μελετήστε τον τρόπο με τον οποίον γίνονται οι αναθέσεις των υπολογισμών στα νήματα χρησιμοποιώντας M= 5, K =3, N = 4 και **#pragma omp for schedule (static/dynamic, chunk) collapse(level)**, με chunk = 2, 3, 5, nthreads = 2, 4, 10, level = 1, 2, 3
Μελετήστε και γράψτε στην αναφορά σας σχετικά με το schedule και το collapse.
Στην οθόνη πρέπει να εμφανίζεται το thread_id που δουλεύει κάθε επαναληπτικό βήμα όπως στην Εικόνα 2.

```
Starting matrix-matrix multiplication example with 4 threads, M= 5, K= 3, N= 4, chunk= 3
Initializing matrices...
Thread=0 did c[i,j] for i=0 , j =0, and k = 0
Thread=0 did c[i,j] for i=0 , j =0, and k = 1
Thread=0 did c[i,j] for i=0 , j =0, and k = 2
Thread=0 did c[i,j] for i=0 , j =1, and k = 0
Thread=0 did c[i,j] for i=0 , j =1, and k = 1
Thread=0 did c[i,j] for i=0 , j =1, and k = 2
```

Εικόνα 2. Πληροφορία σχετικά με το πιο νήμα υπολογίζει το $a_{ik} * b_{kj}$ από το άθροισμα $c_{ij} = \sum_{k=0}^{\cdot\cdot} a_{ik} * b_{kj}$

Δημιουργήστε πίνακες για όλες τις παραπάνω περιπτώσεις που να δείχνουν αυτό το work load, όπως στον Πίνακας 1.

Thread_id for all k (for each I,j)	j			
i	0 0 0	0 0 0	0 0 0	0 0 0
	0 0 0	0 0 0	0 0 0	0 0 0
	0 0 0	0 0 0	0 0 0	0 0 0
	1 1 1	1 1 1	1 1 1	1 1 1
	1 1 1	1 1 1	1 1 1	1 1 1

Πίνακας 1. Οι 3 αριθμοί σε κάθε κελί δείχνουν ποιο νήμα υπολόγισε το $a_{ik} * b_{kj}$ από το άθροισμα $c_{ij} = \sum_{k=0}^{\cdot\cdot} a_{ik} * b_{kj}$, με M=5, K=3, N=4, nthreads = 4, chunk=3, schedule = static, collapse_level= 1.

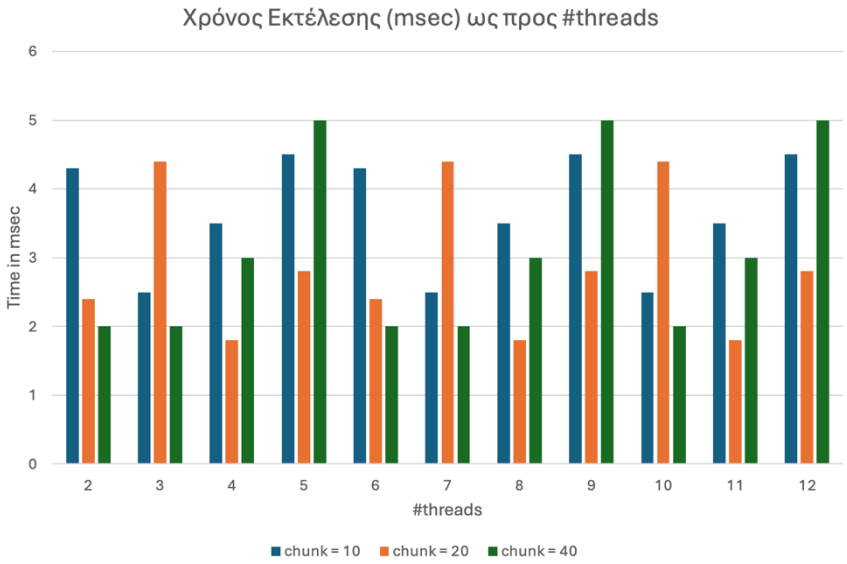
Β) (80) Μελετήστε το πως επηρεάζουν τον χρόνο εκτέλεσης οι διαφορετικές αναθέσεις. Αφού τελειώστε την μελέτη στο **Α)** βάλτε σε σχόλια τα printf που αφορούν τα βήματα που κάνουν threads και αφήστε μόνο τα printf εκτός της παράλληλης διαδικασίας που εμφανίζουν χρόνο και πλήθος επαναλήψεων. Ορίστε M=50, K=10000 (ή όσο μπορεί ο υπολογιστής σας) και N=20, schedule = dynamic και τρέξτε για nthreads = 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 (ή όσα διαθέτει ο υπολογιστής σας), chunk = 10, 20, 40 , level = 1, 2, 3.
Τι παρατηρείτε όσον αφορά :
1) τους χρόνους των threads, τον συνολικό παράλληλο χρόνο και τον χρόνο εκτέλεσης του προγράμματος; Υπάρχει κάποια σχέση μεταξύ τους και ποια;
2) αφού μετρήστε και τον χρόνο από το σειριακό πρόγραμμα (για τα ίδια M,K,N) υπολογίστε χρονοβελτίωση και απόδοση.
Δημιουργήστε πίνακες με την πληροφορία όπως τον Πίνακας 2

Nthreads \ chunk	10	20	40
2	Tp, Sp, Ep	Tp, Sp, Ep	Tp, Sp, Ep
3	Tp, Sp, Ep	Tp, Sp, Ep	Tp, Sp, Ep
4	Tp, Sp, Ep	Tp, Sp, Ep	Tp, Sp, Ep
5	Tp, Sp, Ep	Tp, Sp, Ep	Tp, Sp, Ep

Πίνακας 2. Οι 3 αριθμοί σε κάθε κελί δείχνουν τον παράλληλο χρόνο, τη χρονοβελτίωση και την απόδοση για M=50, K=10000, N=20 και collapse_level= ……..

3) συγκεντρώστε μόνο τους χρόνους εκτέλεσης και φτιάξτε γραφικές παραστάσεις (3 συνολικά, μια για κάθε level) όπως στην Εικόνα 3 για να οπτικοποιήσετε τα δεδομένα σας.

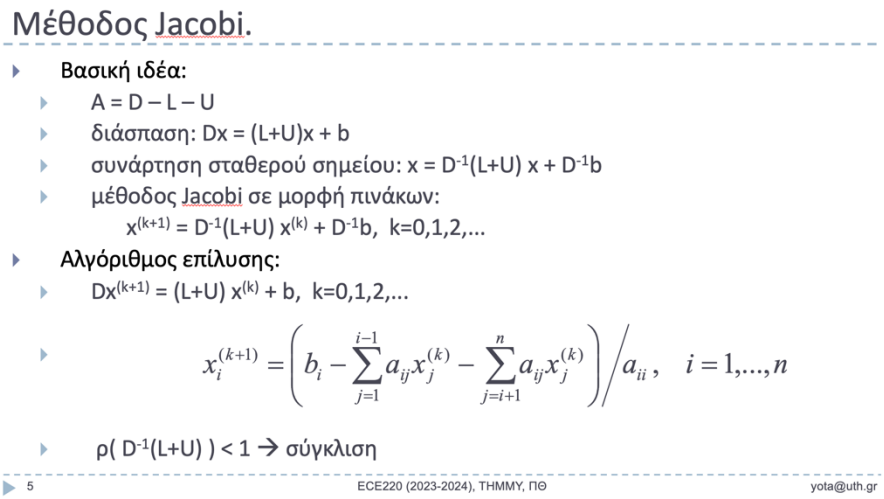
Μπορείτε να κάνετε κάτι αντίστοιχο για τη χρονοβελτίωση και την απόδοση;



Εικόνα 3. Χρόνος εκτέλεσης για level = ….

Άσκηση 3. (240)
Γράψτε συνάρτηση σε C (jacobi_par.c) με χρήση OpenMP, που υλοποιεί την μέθοδο Jacobi (Εικόνα 4) για τη λύση του συστήματος

$$Ax = b$$



Εικόνα 4. Συνοπτική περιγραφή της επαναληπτικής μεθόδου Jacobi για την αριθμητική επίλυση συστήματος γραμμικών εξισώσεων.

Η συνάρτηση:
a) (10) θα δέχεται ορίσματα τον πίνακα A, το διάνυσμα b, τη διάσταση του συστήματος N, το μέγιστο πλήθος επαναλήψεων maxIter, ανοχή στο σφάλμα tol και θα επιστρέφει τη λύση στο διάνυσμα x και τον αριθμό των επαναλήψεων που πραγματοποίησε ενώ θα ορίζεται/καλείται ως jacobi_par (A, b, N, maxIter, tol, x).
b) (25) σε κάθε επανάληψη θα υπολογίζει σωστά την νέα προσέγγιση

c) (25) θα ελέγχει την max-norm του υπολοίπου και αν αυτή είναι μικρότερη της ανοχής tol η επαναληπτική διαδικασία θα τερματίζει.
Υπενθυμίζεται ότι το υπόλοιπο ορίζεται ως $res^{(k+1)} = b-Ax^{(k+1)}$ και η max-norm ενός διανύσματος y ορίζεται ως $\|y\|_\infty = \max_i |y_i|$.
(d) (25) επίσης σε κάθε επανάληψη θα υπολογίζει την max-norm της διαφορά των 2 τελευταίων προσεγγίσεων δηλ. $\left\|x^{(k+1)} - x^{(k)}\right\|_\infty$ και θα εμφανίζει στην οθόνη τα κατάλληλα σχόλια π.χ.,

iter = ***, residual = ***, difference = ***

Γράψτε ένα πρόγραμμα (ask3.c) σε C με OpenMP που να λύνει το παρακάτω σύστημα με την Jacobi.

$$Ax = b \Leftrightarrow \begin{bmatrix} 2 & -1 & \\ -1 & \ddots & -1 \\ & -1 & 2 \end{bmatrix} x = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ N+1 \end{bmatrix}$$

όπου N η διάσταση του συστήματος.
Το πρόγραμμά σας θα πρέπει να :
e) (10) δέχεται σαν ορίσματα το N, maxIter και το tol, να ορίζει τον A και το b όπως παραπάνω και το $x^{(0)} = [0,\ldots,0]^T$. Θα το καλείτε στο command line: ./ask3 100 20 .00005, όπου N=100, maxIter=20, tol=.00005
f) (10) να ξεκινά το παράλληλο περιβάλλον εμφανίζοντας τα κατάλληλα σχόλια (διάσταση συστήματος, μέγιστο πλήθος επαναλήψεων, ανοχή και αριθμό νημάτων) και καλεί την jacobi_par
g) (25) να υπολογίζει την max-norm του σφάλματος της υπολογισθείσας προσέγγισης, όπου σφάλμα $x^* - \tilde{x}$ και να εμφανίζει κατάλληλα σχόλια στην οθόνη, .π.χ.

To sflama ths proseggishs apo th JACOBI einai error = ****

h) (20) να μετρά και να εμφανίζει τον χρόνο εκτέλεσης της jacobi_par με κατάλληλα σχόλια π.χ.

H JACOBI xreiasthke time = *** sec

i) (25) Εκτελέστε το πρόγραμμά σας για διαφορετικό πλήθος νημάτων (1, 2, 4, 5, 8, 10, 12, 16, 20, 32) και καταγράψτε τους χρόνους εκτέλεσης.
j) (40) Δοκιμάστε εμφωλευμένο παραλληλισμό στα 2 for του υπολογισμού της κάθε νέας προσέγγισης. Στην jacobi_par, επαναλάβετε τις εκτελέσεις και καταγράψτε τους χρόνους.
k) (25) Στην αναφορά σας προσθέστε ένα πίνακα, στον οποίον θα φαίνεται ο αριθμός των νημάτων και οι αντίστοιχοι χρόνοι εκτέλεσης στον υπολογιστή σας, όπως στον Πίνακας 3. Σχολιάστε τα αποτελέσματα;

Nthreads \ Parallelism	Simple	Nested
2	….	….
4	….	….
5	….	….
….	….	….

Πίνακας 3. Χρόνοι εκτέλεσης για απλό και εμφωλευμένο παραλληλισμό.

Υπόδειξη: Η λύση του συστήματος είναι το διάνυσμα [1 2 3 … N]. Χρησιμοποιήστε το μεγαλύτερο δυνατό N, π.χ., $N=10^6$ και maxIter = 200.