ECE445: ΠΑΡΑΛΛΗΛΟΙ ΚΑΙ ΔΙΚΤΥΑΚΟΙ ΥΠΟΛΟΓΙΣΜΟΙ
Χειμερινό Εξάμηνο 2024-2025

Εργασία 3:
Προγραμματισμός με MPI
(*Ημερομηνία Παράδοσης: Κυριακή 19.01.2025, 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 → Cross-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.

Παραδείγματα κώδικα θα βρείτε στο .zip που βρίσκεται στο eclass στο φάκελο « Έγγραφα → Αρχικος Κατάλογος → Open MPI material → MPI_examples.zip »

Άσκηση 1. (40)
Γράψτε ένα πρόγραμμα σε C (ask1.c) που υλοποιεί παράλληλο προγραμματισμό με OpenMPI με σκοπό να μετρήσει το κόστος της point2point επικοινωνίας για έναν αριθμό (int, float, double) και καλώντας τις κατάλληλες συναρτήσεις θα εμφανίζει στην οθόνη :
α) (4) το όνομα του «processor» στον οποίο τρέχει το πρόγραμμά σας και πλήθος διεργασιών/tasks που συμμετέχουν στην εκτέλεση (από ένα task).
β) (4) από το master task θα εμφανίζεται στην οθόνη το μήνυμα «Hello. This is the master node. ». Τα υπόλοιπα tasks θα εμφανίζουν στην οθόνη το μήνυμα: «Hello. This is node *.» , όπου * = το id τους
γ) (10) μόνο τα δύο πρώτα tasks θα συμμετέχουν στη διαδικασία μέτρηση της P2P επικοινωνίας, τα υπόλοιπα θα μένουν ανενεργά. Στη συγκεκριμένη διαδικασία θα μετρήσετε τον χρόνο που χρειάζεται για την αποστολή

ενός ακεραίου. Όπως συνήθως, όταν μετράμε χρόνο μέσα στα προγράμματά μας, μετράμε τη διαδικασία πολλές φορές π.χ., 1000 και παίρνουμε το μέσο όρο.
δ) (5) επαναλάβετε τη διαδικασία για αριθμό τύπου float.
ε) (5) επαναλάβετε τη διαδικασία για αριθμό τύπου double.
στ) (2) το main task θα εμφανίσει τον χρόνο εκτέλεσης στην οθόνη.
ζ) (10) μέσα στην αναφορά σας προσθέστε έναν πίνακα (όπως τον Πίνακας 1), στον οποίον θα φαίνεται ο αριθμός των tasks και ο αντίστοιχος χρόνος εκτέλεσης στον υπολογιστή σας, καθώς και screenshot από την εκτέλεση του προγράμματός σας.

Number of tasks	int	float	double
Time (sec)			

Πίνακας 1.Χρόνος εκτέλεσης Point2Point επικοινωνίας.

Άσκηση 2. (40)
Εργαστείτε όπως στην άσκηση 1 και γράψτε ένα πρόγραμμα σε C (ask2.c) που υλοποιεί παράλληλο προγραμματισμό με OpenMPI με σκοπό να μετρήσει το κόστος της εκπομπής ενός ακεραίου (1 int) και καλώντας τις κατάλληλες συναρτήσεις θα εμφανίζει στην οθόνη :
α) (4) το όνομα του «processor» στον οποίο τρέχει το πρόγραμμά σας και πλήθος διεργασιών/tasks που συμμετέχουν στην εκτέλεση (από ένα task).
β) (4) από το master task θα εμφανίζεται στην οθόνη το μήνυμα «Hello. This is the master node. ». Τα υπόλοιπα tasks θα εμφανίζουν στην οθόνη το μήνυμα: «Hello. This is node *.» , όπου * = το id τους
γ) (10) χρησιμοποιήστε την κατάλληλη συνάρτηση MPI για να εκτελέσετε broadcast ενός ακεραίου από τον MASTER node σε όλους τους υπόλοιπους και μετρήστε τον χρόνο.
δ) (10) υλοποιήστε το broadcast ενός ακεραίου αριθμού από τον MASTER προς όλους τους άλλους κόμβους και μετρήστε χρόνο.
Όπως συνήθως, όταν μετράμε χρόνο μέσα στα προγράμματά μας, μετράμε τη διαδικασία πολλές φορές π.χ., 1000 και παίρνουμε το μέσο όρο. Εφαρμόστε το και στο γ) και στο δ).
ε) (2) το main task θα εμφανίσει τον χρόνο εκτέλεσης στην οθόνη.
στ) (5) μέσα στην αναφορά σας προσθέστε έναν πίνακα (όπως τον Πίνακας 2**Error! Reference source not found.**), στον οποίον θα φαίνεται ο αριθμός των tasks και ο αντίστοιχος χρόνος εκτέλεσης στον υπολογιστή σας για κάθε περίπτωση, καθώς και screenshot από την εκτέλεση του προγράμματός σας.

Μπορείτε να χρησιμοποιείτε το option *--oversubscribe* για να χρησιμοποιείτε περισσότερους nodes από ότι φυσικά υποστηρίζει το μηχάνημά σας, π.χ.:
My_prompt >> mpirun -np 8 --oversubscribe ./my_mpi_prog

Number of tasks		2	4	6	8	10	12	20
Time (sec)	MPI function							
	My Implementation							

Πίνακας 2. Χρόνος Εκτέλεσης εκπομπής

Άσκηση 3. (250)
Γράψτε μια MPI/C συνάρτηση **(40)** rowMVMult που θα υλοποιεί πολλαπλασιασμό πίνακα με διάνυσμα (A*b) όπου ο πίνακας A (nxn) είναι αποθηκευμένος κατά μπλοκ γραμμών στις μνήμες των διεργασιών/tasks (p σε πλήθος) και το διάνυσμα b (nx1) θα είναι κι αυτό διαμοιρασμένο στις p tasks.
Η συνάρτηση θα καλείται τοπικά από κάθε task ως rowMVMult(n, localA, localb, localy, comm), όπου localy το αποτέλεσμα της πράξης τοπικά, και comm ο μεταβιβαστής/communicator που συμμετέχουν οι p tasks και περνά σαν όρισμα στη συνάρτηση από το κυρίως πρόγραμμα.

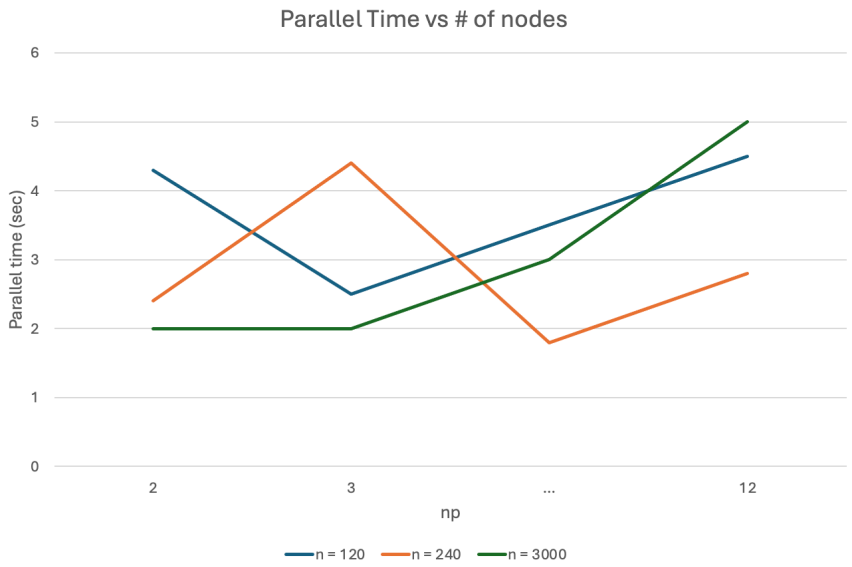
Για να ελέγξετε τη συνάρτησή σας, γράψτε ένα κύριο πρόγραμμα σε MPI/C (ask3.c) **(60)** που να υπολογίζει τον πολλαπλασιασμό πίνακα με διάνυσμα, A*b=y, με A (nxn) και b (nx1) και y(nx1).
Υποθέστε τόσο στο κύριο πρόγραμμα όσο και. στη συνάρτηση ότι το p διαιρεί ακριβώς το n.
Το πρόγραμμά σας θα πρέπει να ικανοποιεί τα παρακάτω:
α) (10) η αρχικοποίηση του πίνακα A και του b θα γίνεται από το MASTER task ως A(i,j) = (i+j)*10.0 και b(i)=10.0.
β) (20) το κάθε κάθε μπλοκ γραμμών του A και στοιχείων του b θα αποστέλλεται στον κατάλληλο node/task.
γ) (20) κάθε task (πλην του MASTER) θα παραλαμβάνει από τον MASTER τα κατάλληλα δεδομένα.
δ) (10) κάθε task θα καλεί την rowMVMult για να υπολογίσει το localy.
ε) (20) μετά τους τοπικούς υπολογισμούς, κάθε task θα στέλνει το localy στον MASTER.
στ) (10) θα καταγράφει τον χρόνο εκτέλεσης του πολλαπλασιασμού A*b=y και θα τον εμφανίζει στην οθόνη με το κατάλληλο μήνυμα (MASTER node).
Όλες οι μεταβλητές θα είναι είτε int είτε double.
Να χρησιμοποιήσετε συλλογικές πράξεις επικοινωνίας μέσα στο κυρίως πρόγραμμα αλλά και μέσα στη συνάρτηση, όπου αυτό είναι εφικτό.

ζ) (60) Τρέξτε το πρόγραμμά σας για n=120, 240, 360, 480, 600, 1200, 1800, 2400, 3000 (ή το μεγαλύτερο δυνατό που αντέχει ο υπολογιστής σας αλλά ≤ 3000) και np = 2, 3, 4, 5, 6, 8, 10, 12 και καταγράψτε τους χρόνους σε ένα πίνακα όπως τον Πίνακας 3

np \ n	120	240	360	……
2				
3				
4				
5				
…..				

Πίνακας 3. Χρόνοι εκτέλεσης για διαφορετικά n και np.

Χρησιμοποιήστε γραφήματα όπως αυτό στην Εικόνα 1 για να οπτικοποιήσετε τα αποτελέσματά σας.



Εικόνα 1.Παράλληλοι χρόνοι ως προς πλήθος tasks για πολλά n.

Άσκηση 4. (250)

Ανατρέξτε στις διαφάνειες *2024.12.10_10.0.pdf* και στο *odd_even.pdf* (απόσπασμα από το βιβλίο των Grama, Karypis et all) για να μελετήσετε τον αλγόριθμο του «Odd-Even Transposition».

Γράψτε **(50)** μια συνάρτηση σε C oddevenser(unsorted, n, sorted) που να υλοποιεί τη σειριακή έκδοση του αλγορίθμου. Προσθέστε σαν όρισμα ότι επιπλέον χρειάζεται να περάσετε στη συνάρτηση.

Γράψτε **(20)** ένα κύριο πρόγραμμα (ask4_ser.c) που θα δέχεται σαν όρισμα στο command line το πλήθος των στοιχείων, n (π.χ., my_prompt >> ask4_ser 100), που θα ταξινομεί και τα οποία θα δημιουργεί με μια γεννήτρια τυχαίων ακεραίων αριθμών (random). Χρησιμοποιήστε τη random σε συνδυασμό με την srandom.

Γράψτε ένα MPI/C πρόγραμμα (ask4_par.c)που θα υλοποιεί την παράλληλη έκδοση του αλγόριθμου.

Το πρόγραμμά σας θα πρέπει να ικανοποιεί τα παρακάτω:

α) **(10)** θα δέχεται σαν όρισμα στο command line το πλήθος των στοιχείων, n (π.χ., my_prompt >> mpirun -np 2 ask4_par 100) που θα ταξινομεί

β) **(20)** κάθε task θα δημιουργεί n/p στοιχεία με μια γεννήτρια τυχαίων ακεραίων αριθμών (συνδυασμός srandom, random).

γ) **(10)** κάθε task θα καλεί τη σειριακή συνάρτηση ταξινόμησης oddevenser για να ταξινομήσει τοπικά τα δικά της στοιχεία

δ) **(100)** μετά την τοπική ταξινόμηση εφαρμόζει κατάλληλα την τεχνική του odd-even για να ανταλλάσσουν τα στοιχεία τους «γειτονικά» tasks έτσι ώστε το task με το μικρότερο id να έχει τα μικρότερα στοιχεία και το task με το μεγαλύτερο id να έχει τα μεγαλύτερα στοιχεία.

Μελετήστε **(40)** το πρόβλημα (με όμοιο τρόπο όπως αυτόν του αποσπάσματος του βιβλίου) και εξηγήστε το πρόγραμμά σας αλλά και τον θεωρητικό παράλληλο χρόνο που θα καταλήξετε.

Πόσες επαναλήψεις χρειάζεται να κάνει το πρόγραμμά πριν τελειώσει η διαδικασία της ταξινόμησης;

Τρέξτε το σειριακό πρόγραμμά σας για διαφορετικά n (π.χ., 100, 200, 300,…..) και μετρήστε το χρόνο εκτέλεσης. Συμπληρώστε πίνακα με τους χρόνους και κάντε τη γραφική παράσταση και παρατηρήστε αν όταν ο χρόνος συμπεριφέρεται όπως πρέπει.

Τρέξτε το παράλληλο πρόγραμμά σας για np 2, 4, 5, 10 κλπ. και για διαφορετικά n (π.χ., 100, 200, 300,…..) και μετρήστε το χρόνο εκτέλεσης. Χρησιμοποιείτε πάντα np τέτοια ώστε να διαιρούν τέλεια το n.

Όμοια με πριν συμπληρώστε πίνακα με τους χρόνους και κάντε τη γραφική παράσταση και παρατηρήστε αν όταν ο παράλληλος χρόνος συμπεριφέρεται όπως αναμένεται.

Τι παρατηρείτε;