


Παράλληλα Συστήματα


Χειμερινό εξάμηνο 2024–2025

#3


Ορισμός διακριτών εργασιών (tasks)

- ▶ Ελλείψεις σε εκδόσεις OpenMP ≤ 2.5
 - ▶ Αυστηρά δομημένη διαδικασία παραλληλοποίησης
 - ▶ Δυνατότητα δυναμικού ορισμού διακριτών εργασιών (tasks)
- 


Ορισμός διακριτών εργασιών (tasks)

- ▶ `#pragma omp task [clause ...] new-line
structured-block`
 - ▶ Στη θέση του clause ορισμένες παράμετροι, όπως και σε άλλες διαταγές
- 


Παράμετροι σε 'task'

- ▶ if
 - ▶ shared
 - ▶ private
- 

Παράμετροι σε 'task' [2]

- ▶ firstprivate
 - ▶ default
 - ▶ untied
- 

Διαταγή 'task'

- ▶ Τρόπος λειτουργίας
 - ▶ Implicit tasks
 - ▶ Explicit tasks
- 

Παράδειγμα

```
#pragma omp parallel default(shared)
{
    #pragma omp single private(p)
    {
        p = headlist;
        num = 0;
        while (p)
        {
            array[num] = p;
            p = next(p);
            num++;
        }
    }
    #pragma omp for private(i)
    for (i=0; i<num; i++)
        process(array[i]);
}
```

Παράδειγμα

```
#pragma omp parallel default(shared)
{
    #pragma omp single private(p)
    {
        p = headlist;
        while (p)
        {
            #pragma omp task
            process(p);
            p = next(p);
        }
    }
}
```


Διαταγή 'taskwait'

- ▶ Δυνατότητες συγχρονισμού
- ▶ `#pragma omp taskwait newline`

Διαταγή 'taskwait'

- ▶ Αναστολή εκτέλεσης της τρέχουσας διακριτής εργασίας
 - Μέχρι όλα τα παιδιά της να ολοκληρώσουν τη δική τους διακριτή εργασία
- ▶ Έλεγχος μόνο για πρώτου επιπέδου (άμεσα) παιδιά
 - Όχι έλεγχος για βαθύτερου επιπέδου παιδιά (εγγόνια)

Παράδειγμα


```
void postorder(node)
{
    if (node->left)
        #pragma omp task
        postorder(node->left);

    if (node->right)
        #pragma omp task
        postorder(node->right);

    #pragma omp taskwait

    process(node->data);
}
```

Παραδείγματα (task)

- task1.c Σειριακό
 - task2.c Παράλληλο
 - task3.c Παράλληλο με single
 - task4.c Παράλληλο με task
 - task5.c Παράλληλο με task (+επανάληψη)
- 

Δήλωση 'reduction'

- ▶ Συγκέντρωση επιμέρους αποτελεσμάτων από κάθε νήμα/επεξεργαστή που είναι αποθηκευμένα σε μία ή περισσότερες μεταβλητές, και η εφαρμογή μίας πράξης σε αυτά
- ▶ Υποστήριξη από OpenMP
 - reduction (operator: list)

Παράδειγμα

- ▶ Εσωτερικό γινόμενο δύο διανυσμάτων a και b (μήκους N)

```
for (i=0; i < N; i++)  
{  
    result = result + (a[i] * b[i]);  
}
```

Παράδειγμα

- ▶ Για τον παράλληλο υπολογισμό των παραπάνω επαναλήψεων, απαιτούνται τα ακόλουθα βήματα:
 - (α) ο διαμοιρασμός των τιμών των διανυσμάτων στα νήματα που έχουν δημιουργηθεί
 - (β) ο πολλαπλασιασμός από κάθε νήμα τιμή προς τιμή των τιμών που του αναλογούν από κάθε διάνυσμα
 - (γ) η πρόσθεση όλων των όρων που προέκυψαν σε μία κοινή μεταβλητή που αναπαριστά και το τελικό αποτέλεσμα.

Παράδειγμα: omp12.c

- ▶ Παράλληλη υλοποίηση με χρήση της δήλωσης reduction
 - #pragma omp for schedule(static,chunk)
reduction(+:result)

Πράξεις σε reduction

- ▶ Σχετικά με την σύνταξη της reduction
 - Παράμετρος list
- ▶ Οι πράξεις που επιτρέπονται να πραγματοποιηθούν σε μια λειτουργία reduction:
 - $+$, $*$, $-$, $\&$, \wedge , $|$, $\&\&$, $||$

Παράδειγμα: omp13.c

- ▶ Αν δεν είχαμε στη διάθεσή μας τον μηχανισμό του reduction, για να υλοποιήσουμε παράλληλα τον υπολογισμό του παραπάνω παραδείγματος (εσωτερικό γινόμενο δύο διανυσμάτων) τότε τι θα κάναμε;
- ▶ Οφέλη από reduction;

Δήλωση 'collapse'

- ▶ Σε μία διαταγή διαμοιρασμού εργασίας for, παρέχεται η δυνατότητα ενιαίου διαμοιρασμού ενός συνόλου επαναλήψεων οι οποίες δηλώνονται στα πλαίσια ενός ή περισσότερων του ενός εμφωλιασμένων for-loops
- ▶ Διευκολύνεται έτσι ο αρτιότερος διαμοιρασμός φόρτου και δεδομένων σε πιο σύνθετα προβλήματα και υπολογισμούς
- ▶ Η δυνατότητα αυτή παρέχεται μέσω της δήλωσης collapse

Δήλωση ‘collapse’

- ▶ Σύνταξη
 - collapse(n)
- ▶ $n \rightarrow$ ο αριθμός των επιπέδων εμφωλιασμού που επιθυμούμε

Παραδείγματα (collapse)

- ▶ For μέσα σε for με 4 νήματα
 - c1.c
- ▶ For μέσα σε for με 8 νήματα
 - c2.c
- ▶ Ενιαίος βρόγχος (με 8 νήματα)
 - c3.c

Παραδείγματα με collapse

- ▶ Αν στο προηγούμενο σύνολο βρόγχων κάνουμε χρήση της δήλωσης `collapse(1)`;
 - c4.c: 4η απόπειρα: 8 νήματα + `collapse(1)`
- ▶ Αν στο προηγούμενο σύνολο βρόγχων κάνουμε χρήση της δήλωσης `collapse(2)`;
 - c5.c: 5η απόπειρα: 8 νήματα + `collapse(2)`

Άλλες δυνατότητες εμφωλισμού

- ▶ Γενικότερες δυνατότητες εμφωλισμένου παραλληλισμού μέσω των οποίων καθίσταται δυνατή η διατύπωση σε παράλληλο μορφή πιο σύνθετων αλγοριθμικών δομών.
- ▶ Μέσα σε μία διαταγή parallel μπορεί κανείς να ορίσει/εμφωλιάσει άλλη διαταγή parallel κ.ο.κ. ως ένα μέγιστο επιτρεπόμενο αριθμό επιπέδων ο οποίος ορίζεται από το σύστημα

Παράδειγμα: omp15.c

```
int main(int argc, char **argv) {  
    omp_set_nested(1);  
    omp_set_num_threads(2);  
    int tid;  
    #pragma omp parallel private(tid) {  
        tid = omp_get_thread_num();  
        printf("Hello 1 !!! from thread %d\n", tid);  
        #pragma omp parallel private(tid) {  
            tid = omp_get_thread_num();  
            printf("Hello 2 !!! from thread %d\n", tid);  
            #pragma omp parallel private(tid) {  
                tid = omp_get_thread_num();  
                printf("Hello 3 !!! from thread %d\n", tid);  
            }  
        }  
    }  
}
```


Ενεργοποίηση δυνατότητας εμφωλιασμένου παραλληλισμού

- ▶ OMP_NESTED

- true

- ▶ omp_set_nested()

Παράδειγμα: false.c

- ▶ Αν η δυνατότητα εμφωλιασμένου παραλληλισμού είναι απενεργοποιημένη) κάθε φορά που μέσα σε μία διαταγή parallel συναντάται μία άλλη διαταγή parallel, για κάθε νήμα που εκτελεί την πρώτη διαταγή εκκινείται μεν τυπικά μία νέα ομάδα νημάτων η οποία ωστόσο αποτελείται μόνο από ένα (το τρέχον) νήμα.

Παράδειγμα: false.c

```
int main(int argc, char **argv) {  
    omp_set_nested(0);  
    omp_set_num_threads(2);  
    int tid;  
    #pragma omp parallel private(tid) {  
        tid = omp_get_thread_num();  
        printf("Hello 1 !!! from thread %d\n", tid);  
        #pragma omp parallel private(tid) {  
            tid = omp_get_thread_num();  
            printf("Hello 2 !!! from thread %d\n", tid);  
            #pragma omp parallel private(tid) {  
                tid = omp_get_thread_num();  
                printf("Hello 3 !!! from thread %d\n", tid);  
            }  
        }  
    }  
}
```

Συγχρονισμός / επικοινωνία νημάτων

- ▶ Δεν παρέχεται από το OpenMP κάποιος ειδικότερος μηχανισμός για την υλοποίηση πιο σύνθετων αναγκών συγχρονισμού/επικοινωνίας μεταξύ των πολλαπλών νημάτων
- ▶ Για την υποστήριξη τέτοιου είδους αναγκών ωστόσο είναι δυνατή η χρήση των γενικότερων συναρτήσεων κλειδώματος (locks) τις οποίες παρέχει.

Άσκηση: pingpong.c

- ▶ Παράδειγμα απλής αναμονής με χρήση των συναρτήσεων κλειδώματος:
 - `omp_set_lock`, `omp_unset_lock`, ...
- ▶ Έστω ότι μας ζητείται να συγχρονίσουμε δύο νήματα έτσι ώστε να τυπώνουν επαναληπτικά μία ακολουθία από εναλλασσόμενες εμφανίσεις των λεκτικών 'PING' και 'PONG'
 - PING PONG PING PONG PING ...

Άλλες συναρτήσεις του OpenMP[1]

- ▶ Πέραν των βασικών συναρτήσεων
 - `omp_set_num_threads()`
 - `omp_get_num_threads()`
 - `omp_get_thread_num()`
 - ...
- ▶ Το OpenMP προσφέρει και έναν ακόμα σημαντικό αριθμό συναρτήσεων

Άλλες συναρτήσεις του OpenMP[2]

- ▶ `int omp_get_max_threads(void);`
 - Επιστρέφει τον μέγιστο αριθμό νημάτων που μπορούν να δοθούν σε μία νέα ομάδα νημάτων ως αποτέλεσμα μιας νέας κλήσης της διαταγής `parallel`
- ▶ `int omp_get_thread_limit(void);`
 - Επιστρέφει τον μέγιστο συνολικά (σε αντιδιαστολή με την προαναφερθείσα) αριθμό νημάτων που είναι διαθέσιμα εξ αρχής στο πρόγραμμα.
- ▶ `int omp_get_num_procs(void);`
 - Επιστρέφει τον αριθμό των διαθέσιμων επεξεργαστών/πυρήνων του συστήματος.
- ▶ `int omp_in_parallel(void);`
 - Επιστρέφει αληθές αν η κλήση έχει γίνει εντός μίας ενεργής παράλληλης περιοχής, ειδάλλως επιστρέφει ψευδές.

Άλλες συναρτήσεις του OpenMP[3]

- ▶ `void omp_set_dynamic(int dyn_threads);`
 - Ενεργοποιεί (`dyn_threads≠0`) ή απενεργοποιεί (`dyn_threads=0`) τη δυνατότητα δυναμικού προσδιορισμού του αριθμού των διαθέσιμων νημάτων, θέτοντας ανάλογα την εσωτερική μεταβλητή ελέγχου `dyn-var`.
- ▶ `int omp_get_dynamic(void);`
 - Επιστρέφει την τιμή της εσωτερικής μεταβλητής ελέγχου (ICV) `dyn-var`, η οποία προσδιορίζει αν η δυνατότητα δυναμικού προσδιορισμού του αριθμού των διαθέσιμων νημάτων είναι ενεργοποιημένη ή απενεργοποιημένη.
- ▶ `void omp_set_nested(int nested);`
 - Ενεργοποιεί (`nested≠0`) ή απενεργοποιεί (`nested=0`) τη δυνατότητα εμφωλιασμένου παραλληλισμού, θέτοντας ανάλογα την εσωτερική μεταβλητή ελέγχου (ICV) `nest-var`.
- ▶ `int omp_get_nested(void);`
 - Επιστρέφει την τιμή της εσωτερικής μεταβλητής ελέγχου (ICV) `nest-var`, η οποία προσδιορίζει αν η δυνατότητα εμφωλιασμένου παραλληλισμού είναι ενεργοποιημένη ή απενεργοποιημένη.

Άλλες συναρτήσεις του OpenMP[4]

- ▶ `void omp_set_max_active_levels(int max_levels);`
 - Περιορίζει τον μέγιστο επιτρεπόμενο αριθμό επιπέδων ενεργών εμφωλιασμένων παράλληλων περιοχών (parallel regions), θέτοντας ανάλογα την εσωτερική μεταβλητή ελέγχου (ICV) `max-active-levels-var`.
- ▶ `int omp_get_max_active_levels(void);`
 - Επιστρέφει την τιμή της εσωτερικής μεταβλητής ελέγχου (ICV) `max-active-levels-var`, η οποία προσδιορίζει τον μέγιστο επιτρεπόμενο αριθμό επιπέδων ενεργών εμφωλιασμένων παράλληλων περιοχών.
- ▶ `int omp_get_level(void);`
 - Επιστρέφει τον αριθμό των εμφωλιασμένων παράλληλων περιοχών από τις οποίες περικλείεται η διακριτή εργασία η οποία έκανε την κλήση.
- ▶ `int omp_get_active_level(void);`
 - Επιστρέφει τον αριθμό των εμφωλιασμένων ενεργών παράλληλων περιοχών από τις οποίες περικλείεται η διακριτή εργασία η οποία έκανε την κλήση.

Άλλες συναρτήσεις του OpenMP[5]

- ▶ `int omp_get_ancestor_thread_num(int level);`
 - Επιστρέφει τον αριθμό του νήματος-προγόνου του τρέχοντος νήματος, σε επίπεδο εμφωλιασμού αυτό που προσδιορίζεται από την παράμετρο `level`.
- ▶ `int omp_get_team_size(int level);`
 - Επιστρέφει το πλήθος των νημάτων της ομάδας του νήματος-προγόνου του τρέχοντος νήματος, σε επίπεδο εμφωλιασμού όπως προσδιορίζεται από την παράμετρο `level`.
- ▶ `double omp_get_wtime(void);`
 - Επιστρέφει μία χρονική ένδειξη (πόσος χρόνος έχει περάσει) σε δευτερόλεπτα από κάποιο πρότερο σταθερό χρονικό σημείο.
- ▶ `double omp_get_wtick(void);`
 - Επιστρέφει την ακρίβεια (πόσος χρόνος μεσολαβεί μεταξύ δύο ticks) του μετρητή/ρολογιού που χρησιμοποιείται από τη συνάρτηση `omp_get_wtime()`.

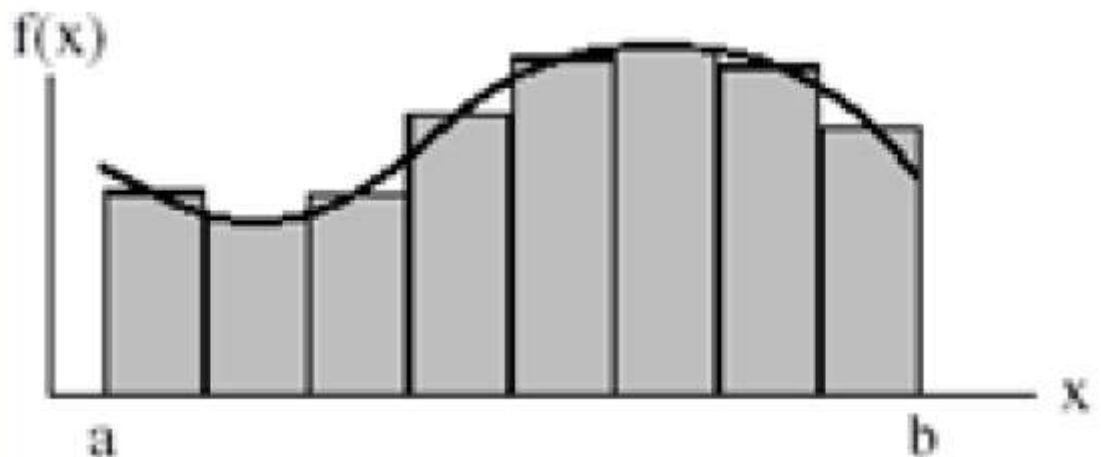
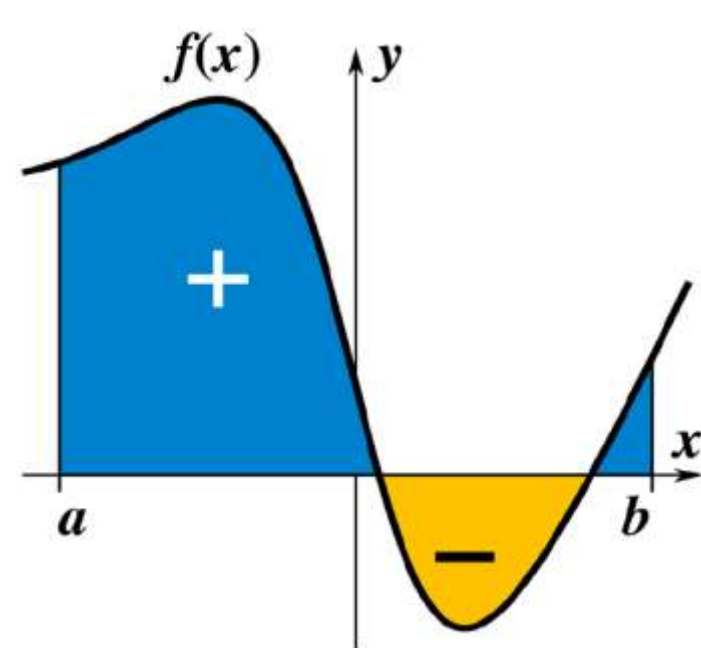
Υπολογισμός του π

- ▶ Ο αριθμός 'π' μπορεί να υπολογιστεί ως η τιμή του ολοκληρώματος

$$\int_0^1 \frac{4}{1+x^2} dx$$

Υπολογισμός του π

- ▶ Το ολοκλήρωμα της συνάρτησης $f(x)$ από το a στο b είναι η επιφάνεια πάνω από τον άξονα x και κάτω από την καμπύλη $y = f(x)$, μείον την επιφάνεια κάτω από τον άξονα x και πάνω από την καμπύλη, για x στο διάστημα $[a,b]$.



$$\int_a^b f(x) dx \approx \sum_{i=1}^n h f(a + (i - 1/2)h)$$

Παράδειγμα: pi-nr.c

- ▶ Παράλληλος υπολογισμός του π χωρίς χρήση του reduction
 - Προστασία της διαμοιραζόμενης μεταβλητής `sum` που κρατάει το σύνολο των αθροισμάτων

Παράδειγμα: omp19.c

- ▶ Χρήση της `omp_get_wtime()` σε δύο σημεία του κώδικα και υπολογισμός της διαφοράς μεταξύ των δύο αυτών χρονικών σημείων
 - => Χρόνος εκτέλεσης

Παράδειγμα: omp20.c

- ▶ Υπολογισμός π με χρήση της δήλωσης `reduction` και κλήσεις στη συνάρτηση `omp_get_wtime` για μέτρηση χρόνου