ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ

ΕΜ 361: Παράλληλοι Υπολογισμοί

Ενότητα #5Α: Λογισμικό, Βασικές Εφαρμογές – OpenMP

Διδάσκων: Χαρμανδάρης Ευάγγελος
ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΩΝ ΜΑΘΗΜΑΤΙΚΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται στην άδεια χρήσης **Creative Commons** και ειδικότερα

Αναφορά – Μη εμπορική Χρήση – Όχι Παράγωγο Έργο 3.0 Ελλάδα

(Attribution – Non Commercial – Non-derivatives 3.0 Greece)

[ή επιλογή ενός άλλου από τους έξι συνδυασμούς]

[και αντικατάσταση λογότυπου άδειας όπου αυτό έχει μπει (σελ. 1, σελ. 2 και τελευταία)]

- Εξαιρείται από την ως άνω άδεια υλικό που περιλαμβάνεται στις διαφάνειες του μαθήματος, και υπόκειται σε άλλου τύπου άδεια χρήσης. Η άδεια χρήσης στην οποία υπόκειται το υλικό αυτό αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Κρήτης**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

ΕΜ 361: Παράλληλοι Υπολογισμοί

Χαρμανδάρης Βαγγέλης, Τμήμα Εφαρμοσμένων Μαθηματικών

Πανεπιστήμιο Κρήτης, Χειμερινό Εξάμηνο 2010/11

Κεφάλαιο 5:
(Α) Λογισμικό, Βασικές Εφαρμογές – OpenMP

- **Παράλληλες Γλώσσες Δεδομένων (Fortran 90, C, C++).**

- **Βιβλιοθήκες Ανταλλαγής Μηνυμάτων (Message Passing Interface, MPI), OpenMP, POSIX Threads.**

- **Παράλληλοι Αλγόριθμοι με Χρήση OpenMP.**

Λογισμικό

Λογισμικό για ανάπτυξη παράλληλων προγραμμάτων:

• Διαθέσιμα πακέτα: γενικής χρήσεως ή εξειδικευμένοι αλγόριθμοι.
-- **Υπέρ:** Εύκολη λύση.
-- **Κατά:** Μη-κατανόηση του αλγόριθμου και του προγράμματος.

• Ανάπτυξη κώδικα από την αρχή
-- **Υπέρ:** Ευελιξία, Προσαρμοστικότητα, Κατανόηση.
-- **Κατά:** Μπορεί να είναι πολύ χρονοβόρα λύση.

• Δυνατότητα χρήσης προγραμμάτων από παράλληλες βιβλιοθήκες: π.χ. χρήση των βιβλιοθηκών BLAS, ScaLAPack κ.α.
-- **Υπέρ:** Οι βιβλιοθήκες περιέχουν πολύ γρήγορους και δοκιμασμένους αλγόριθμους.
-- **Κατά:** Δύσκολη κατανόηση του αλγόριθμου και του προγράμματος.

Λογισμικό Παράλληλου Προγραμματισμού

• Διεπιφάνεια Προγραμματισμού και Εφαρμογών (**Application Programming Interface**)

- Παράλληλες Γλώσσες Δεδομένων (Parallel Data Languages): Πράξεις μεταξύ διανυσμάτων, πινάκων (π.χ. Fortran 90, high Performance Fortran)

- Μοντέλο Δεσμών (Threads model): Πολλαπλές διεργασίες που επικοινωνούν μέσω του κυρίως προγράμματος.

- Μοντέλο Μεταφοράς Μηνύματος (Message Passing model): οι διεργασίες επικοινωνούν με ανταλλαγές μηνυμάτων.

•Αυτόματος Παραλληλισμός

- Αυτόματη μετατροπή του κώδικα. Συνήθως μετατροπή απλών επαναληπτικών διαδικασιών (loops).
- Εύκολοι στη χρήση αλλά με περιορισμένες δυνατότητες.

Παράλληλες Γλώσσες Δεδομένων

- Συνήθως εμπεριέχουν σύνολο εντολών προέκτασης της κανονικής (σειριακή) γλώσσας.
- **Fortran 90 και 95** (F90, F95):
  - Πράξεις μεταξύ διανυσμάτων, πινάκων

Fortran

```
Real  x(100), y(100)
…..
Do i = 1, 100
  y(i) = y(i)*x(i)
End do
```

F90

```
Real  ::  x(100), y(100)
….
y = y*x*z
```

- **High Performance Fortran** (HPF):
  - Περιέχει ειδικές εντολές (compiler directives) που λένε στον compiler πώς να κατανέμει δεδομένα μεταξύ των επεξεργαστών.

Παράλληλες Γλώσσες Δεδομένων

Πολλές παραλλαγές/επεκτάσεις της C .

Οι περισσότερες από αυτές έχουν αναπτυχθεί για συγκεκριμένες αρχιτεκτονικές υπολογιστικών συστημάτων .

• **C***: developed in ~ 1990 by Thinking Machines Corporation, for the connection machine (SIMD algorithms).

• **Sequent C**: parallel programming under DYNIX (a version of UNIX).

• **nCUBE C^{+}**: επέκταση της C κατά την οποία ένα πρόγραμμα τρέχει ολόκληρο σε κάθε επεξεργαστή. Μοντέλο SPMD (Single Pogram Multiple Data).

• **OCCAM**: developd in ~ 1978 by Immos Limited for the Transputer series of processor.

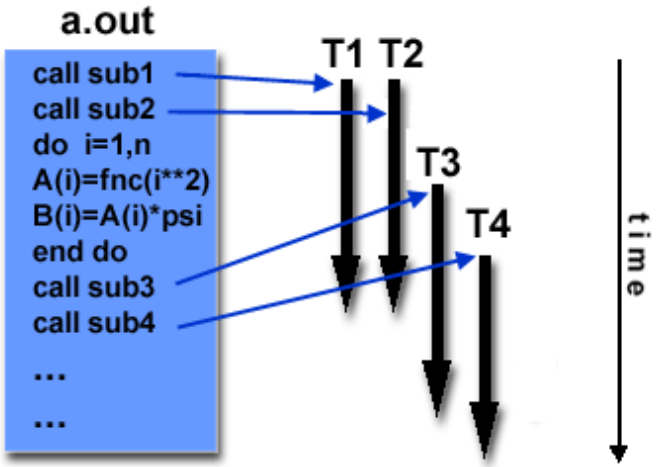
• **C-LINDA**: consists of several operations that work on tople space, a speical form of shared memory.

Μοντέλο Δεσμών (Threads model)

• Στο μοντέλο δεσμών παράλληλου προγραμματισμού μια διεργασία μπορεί να έχει πολλαπλούς, ταυτόχρονους δρόμους εκτέλεσης (execution paths).

• **Παράδειγμα:** κύριο πρόγραμμα *main.exe* που περιλαμβάνει σειρά από υπορουτίνες.

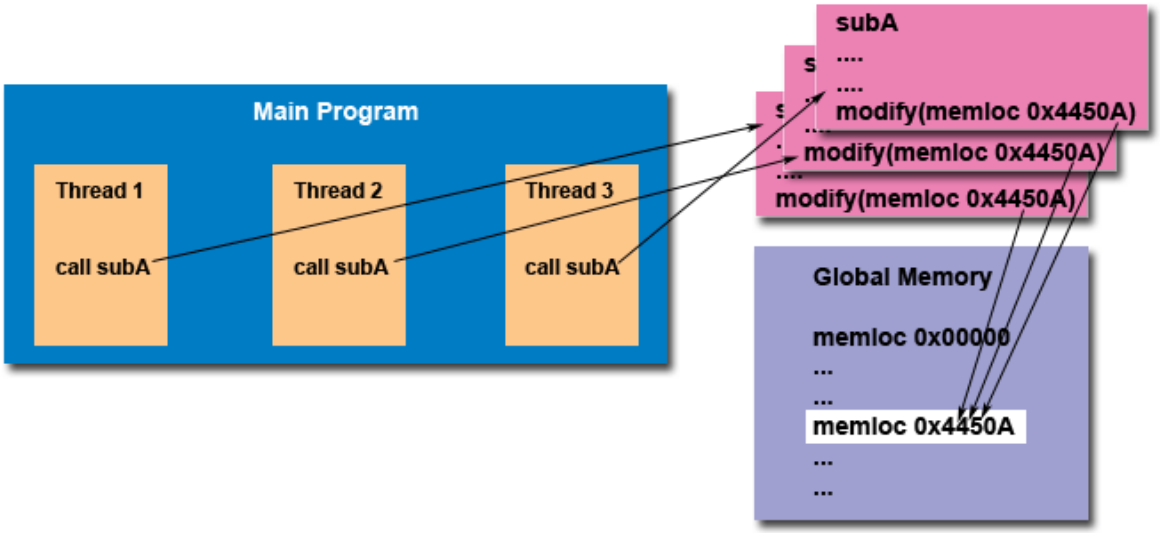
- Μετά από εκτέλεση ενός σειριακού μέρους το *main.exe* δημιουργεί μια σειρά από εργασίες (δέσμες) οι οποίες μπορούν να εκτελεστούν ταυτόχρονα.
- Κάθε εργασία έχει τοπικά δεδομένα (local data) και μοιράζεται τα δεδομένα του *main.exe*. Κατόπιν ανατίθεται σε διαφορετικό επεξεργαστή.
- Οι δέσμες επικοινωνούν μεταξύ τους μέσα από την γενική (global) κοινή μνήμη.
- Το *main.exe* είναι υπεύθυνο για την δημιουργία των δεσμών και την παροχή των κοινών δεδομένων που χρειάζεται κάθε δέσμη.



Μοντέλο Δεσμών (Threads model)

- Από πλευράς προγραμματισμού το μοντέλο δεσμών συνήθως περιλαμβάνει
  - Βιβλιοθήκη από υπορουτίνες που καλούνται από το κυρίως πρόγραμμα.
  - Σετ από οδηγίες για τον compiler που εμπεριέχονται στον παράλληλο κώδικα (compiler directives).

- **Κατάλληλο για συστήματα μοιραζόμενης – κοινής μνήμης.**



Μοντέλο Δεσμών (Threads model)

Διαφορετικές εφαρμογές του μοντέλου δεσμών:

POSIX Δέσμες:

- Βασίζεται σε συναρτήσεις βιβλιοθήκης.
- Διαθέσιμο μόνο για την C.
- Συχνά αναφέρεται και ως Pthreads.
- Προσφέρει πολύ εξειδικευμένο παραλληλισμό. Αρκετά πολύπλοκο στην χρήση.

OpenMP:

- Βασίζεται σε compiler directives.
- Διαθέσιμο τόσο για C/C++ και Fortran.
- Μπορεί να είναι πολύ εύκολο στην χρήση, ειδικά για παραλληλισμό επαναληπτικών διαδικασιών (loops).

• Προσοχή: το μοντέλο δεσμών είναι αρκετά ισχυρό για shared memory machines αλλά όχι για άλλες αρχιτεκτονικές. Δύσκολη η μεταφορά του παράλληλου κώδικα σε διαφορετικά συστήματα.

Μοντέλο Μεταφοράς Μηνύματος (Message Passing)

• Το μοντέλο μεταφοράς μηνύματος έχει τα ακόλουθα χαρακτηριστικά:

➢Υπάρχει σύνολο από διεργασίες οι οποίες χρησιμοποιούν την δικιά τους μνήμη κατά τη διάρκεια των υπολογισμών.

➢ Οι διεργασίες ανταλλάσουν δεδομένα στέλνοντας και λαμβάνοντας μηνύματα.

➢ Οι διαδικασία ανταλλαγής συνήθως χρειάζεται συνεργασία μεταξύ των διεργασιών. Παράδειγμα η αποστολή μηνύματος πρέπει συνδυάζεται με την λήψη του από άλλη διεργασία.

• Η επικοινωνία γίνεται μέσα από βιβλιοθήκες ανταλλαγής μηνυμάτων (π.χ. MPI, PVM) τύπου *SEND* και *RECEIVE*.

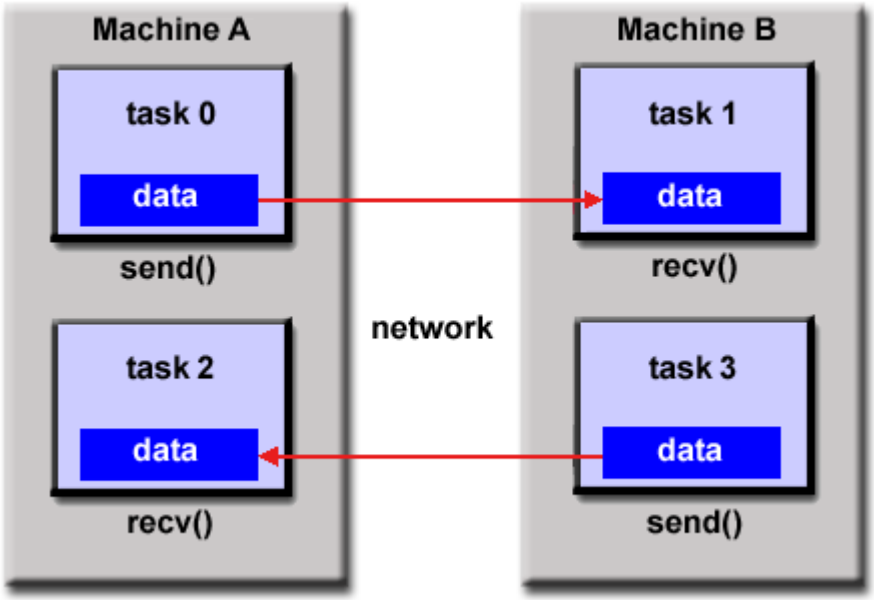
• Η βιβλιοθήκη **MPI (Message Passing Interface)** είναι το διεθνές στάνταρτ.

Μοντέλο Μεταφοράς Μηνύματος (Message Passing)

• Από πλευράς προγραμματισμού ο χρήστης είναι απολύτως υπεύθυνος για τον προσδιορισμό της παράλληλης διαδικασίας.

• **Κατάλληλο τόσο για συστήματα κοινής αλλά και κατανεμημένης μνήμης.**

• Οι κώδικες επεκτείνονται και μεταφέρονται σε συστήματα διαφορετικής αρχιτεκτονικής σχετικά εύκολα.



Αυτόματος Παραλληλισμός

• Παραλληλισμός σειριακού κώδικα μέσω αυτόματων παράλληλων μεταγλωττιστών (**parallel compilers**).

• Εύκολοι στη χρήση: απαιτείται λίγη έως καθόλου δουλεία από τον χρήστη.

• Αυτόματος παραλληλισμός επαναληπτικών διαδικασιών (loops): ανάθεση κομματιών της επαναληπτικής διαδικασίας σε διαφορετικούς επεξεργαστές.

• Μειονεκτήματα:

- Παραλληλισμός διεργασιών δεν είναι εφικτός.
- Δεν είναι ευέλικτος.
- Μπορεί να παράγει λάθος αποτελέσματα.

• Γενικά τα αποτελέσματα είναι πολύ φτωχά: η πολυπλοκότητα των επαναληπτικών διαδικασιών δεν επιτρέπει την αυτόματη παραληλλοποίηση τους.

Κεφάλαιο 5: Open Multi-Processing (OpenMP)

Βασική ιδέα: Χρησιμοποιώντας ένα σύνολο ειδικών εντολών παραλληλίζουμε μέρη του υπάρχοντα σειριακού κώδικα.

Θα δούμε πιο αναλυτικά:

- **Τι είναι το OpenMP; Εισαγωγικά.**
- **Ένα απλό πρόγραμμα σε OpenMP.**
- **Παραλληλισμός επαναληπτικών διαδικασιών (loops) χρησιμοποιώντας OpenMP.**

Τι Είναι το OpenMP;

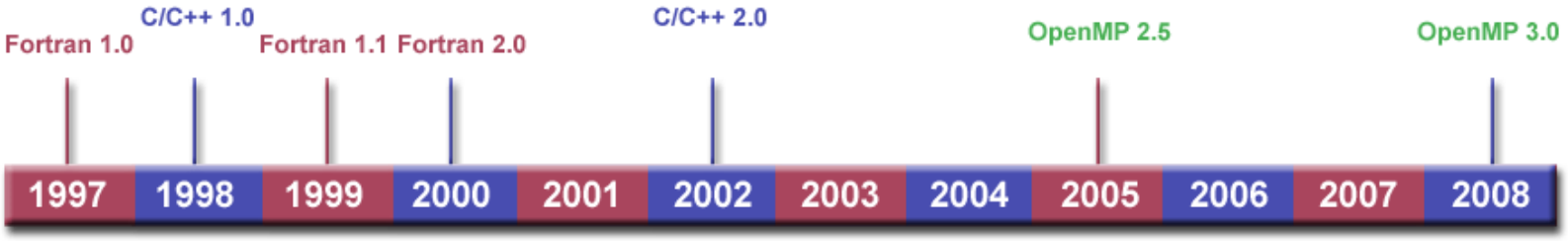
• Το OpenMP είναι μια Διεπιφάνεια Προγραμματισμού και Εφαρμογών (**Application Programming Interface**) παραλληλισμού για συστήματα κοινής μνήμης (**shared memory systems**). Αποτελείται από:

- Compiler Directives
- Runtime Library Routines
- Environmental Variables

• Χαρακτηριστικά του OpenMP: Standard για τα περισσότερα συστήματα κοινής μνήμης, Εύκολο στη χρήση, Αποτελεσματικότητα, Φορητότιτα.

• Σχεδιασμένο το για C/C++ όσο και Fortran/ Fortran 90.

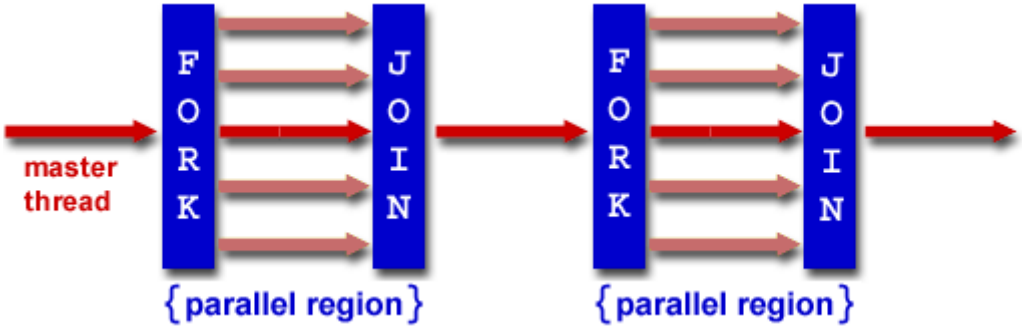
• Ιστορία και Εξέλιξη:



Τι Είναι το OpenMP;

• Το OpenMP βασίζεται στην ύπαρξη πολλαπλών δεσμών (**multi-threaded**) σε διεργασίες κοινής μνήμης. Το πρόγραμμα χωρίζεται σε παράλληλα και σειριακά μέρη.

• Κάθε OpenMP πρόγραμμα ξεκινά με μια αρχική διεργασία (**master thread**) η οποία εκτελείται σειριακά έως ότου φτάσουμε στην πρώτη παράλληλη περιοχή.



• **Compiler Directive based**: ο παραλληλισμός επιτυγχάνεται με την χρήση κατάλληλων οδηγιών που εμπεριέχονται στον κώδικα (C/C++ ή Fortran).

Παράδειγμα: 'Hello world' πρόγραμμα

Απλό πρόγραμμα σε Fortran

```
Program Hallo

Integer  :: NTHREADS, TID, OMP_GET_NUM_THREADS, OMP_GET_THREAD_NUM

!Fork a team of threads with each thread having a private TID variable
!$OMP PARALLEL PRIVATE(TID)

! Obtain and print thread id
TID = OMP_GET_THREAD_NUM()
PRINT *, 'Hello World from thread = ', TID

! Only master thread does this
IF (TID == 0) THEN
  NTHREADS = OMP_GET_NUM_THREADS()
  PRINT *, 'Number of threads = ', NTHREADS
END IF

! All threads join master thread and disband
!$OMP END PARALLEL

Stop
End
```

Παράδειγμα: 'Hello world' πρόγραμμα

Απλό πρόγραμμα σε C

```
#include <omp.h>
 main () {

int nthreads, tid;

/* Fork a team of threads with each thread having a private tid variable */
#pragma omp parallel private(tid)
  {
    /* Obtain and print thread id */
    tid = omp_get_thread_num();
    printf("Hello World from thread = %d\n", tid);

    /* Only master thread does this */
    if (tid == 0)
    {
      nthreads = omp_get_num_threads();
      printf("Number of threads = %d\n", nthreads);
    }
  }  /* All threads join master thread and terminate */

}
```

Περιγραφή Απλού Προγράμματος

Η διάταξη των εντολών/οδηγιών OpenMP (compiler directives) είναι:

- **Fortran:** *!$OMP directive.name [clause]*
  - Παράδειγμα: *!$OMP PARALLEL DEFAULT(SHARED) PRIVATE(BETA,PI)*
  - Το end directive είναι προαιρετικό αλλά βοηθά στον έλεγχο του κώδικα.
  - Case Insensitive.
  - Παράδειγμα:

```
!$OMP directive
   [ structured block of code ]
!$OMP end directive
```

- **C/C++:** *#pragma omp directive.name [clause]*
  - Παράδειγμα: *#pragma omp parallel default(shared) private(beta,pi)*
  - Case Sensitive.

Γενικά: Οι compilers εμπεριέχουν μια μεταβλητή η οποία κατά τη μετάφραση ενεργοποιεί όλες τις OpenMP εντολές (OpenMP directives).

Δημιουργία Παράλληλης Περιοχής

Η παράλληλη περιοχή (parallel region) είναι ένα κομμάτι του κώδικα που μπορεί να εκτελεστεί από πολλαπλές δέσμες (threads). Είναι η σημαντικότερη εντολή του OpenMP.

Μόλις το πρόγραμμα συναντά μια εντολή *parallel* δημιουργεί πολλαπλές threads οι οποίες εκτελούν τον ίδιο κώδικα.

Format:

- Fortran:

```
!$OMP PARALLEL [clause ...]
IF (scalar_logical_expression)
PRIVATE (list)
SHARED (list)
DEFAULT (PRIVATE | FIRSTPRIVATE | SHARED | NONE)
FIRSTPRIVATE (list)
REDUCTION (operator: list)
COPYIN (list)
NUM_THREADS (scalar-integer-expression)
   block
!$OMP  END PARALLEL
```

Παράλληλη Περιοχή

Format:

- C:

```
#pragma omp parallel [clause ...]  newline
if (scalar_expression)
private (list)
shared (list)
default (shared | none)
firstprivate (list)
reduction (operator: list)
copyin (list)
num_threads (integer-expression)
  structured_block
```

• Η μεταβλητή περιβάλλοντος (environmental variable) OMP_NUM_THREADS καθορίζει πόσες threads θα χρησιμοποιηθούν. Συνήθως ο αριθμός των threads είναι ο αριθμός των επεξεργαστών ου συστήματος. Οι threads αριθμούνται από 0 ως P-1.

•**omp_get_thread_num()**: επιστρέφει τον αριθμό της συγκεκριμένης thread.

•**omp_get_num_threads()**: επιστρέφει τον συνολικό αριθμό των threads.

Παραλληλισμός Επαναληπτικών Διαδικασιών

Παραλληλισμός επαναληπτικών διαδικασιών (loops) με την εντολή *!omp do*.

Format:

- C:

```
#pragma omp for [clause ...]  newline
schedule (type [,chunk])
 ……
  for_loop
```

- Fortran:

```
!$omp do [clause ...]
                SCHEDULE (type [,chunk])
                 ……
   do_loop
!$omp end do
```

• **schedule**: περιγράφει πως κατανέμονται οι επαναλήψεις του loop. Ο τύπος (type) μπορεί να είναι στατικός (static) ή δυναμικός (dynamic). Κάθε κομμάτι έχει μέγεθος chunk. Αν το μέγεθος δεν ορίζεται οι επαναλήψεις ισοκατανέμονται (αν αυτό είναι δυνατόν) σε κάθε επεξεργαστή.

Παράδειγμα: Παραλληλισμός Loop με OpenMP

```
#include <omp.h>
#define CHUNKSIZE 100
#define N 1000
main () {

int i, chunk;
float a[N], b[N], c[N];

/* Some initializations */
for (i=0; i < N; i++)
  a[i] = b[i] = i * 1.0;
chunk = CHUNKSIZE;

#pragma omp parallel shared(a,b,c,chunk) private(i)
  {
   #pragma omp for schedule(dynamic,chunk) nowait
   for (i=0; i < N; i++)
      c[i] = a[i] + b[i];
   } /* end of parallel section */

}
```

Περισσότερες Εντολές OpenMP

- **OMP Section:** Χρησιμοποιείται για παραλληλισμό μη επαναληπτικής διαδικασίας (όχι loop). Δηλώνει ότι η ακόλουθη περιοχή θα κατανεμηθεί σε διαφορετικές threads. Κάθε section εκτελείται από άλλη thread.

Format:

❑ **C:**

```
#pragma omp sections [clause ...]  newline
 {
   #pragma omp section newline
          block of code
  #pragma omp section newline
         block of code
  }
```

❑ **Fortran:**

```
!$OMP SECTIONS [clause ...]
   !$OMP SECTION
          block of code
   !$OMP SECTION
          block of code
!$OMP END SECTIONS
```

Περισσότερες Εντολές OpenMP

- **OMP Master:** Δηλώνει ότι η ακόλουθη περιοχή θα εκτελεστεί μόνο από την κεντρική thread (κεντρικό επεξεργαστή).
  C: *#pragma omp master*
  Fortran: *!$OMP MASTER*

- **OMP Single:** Δηλώνει ότι η ακόλουθη περιοχή θα εκτελεστεί μόνο από μία (οποιαδήποτε) thread.
  C: *#pragma omp single*
  Fortran: *!$OMP SINGLE*

- **OMP PARALLEL DO/ parallel for:** Συνδυασμός 2 εντολών OpenMP. Ισοδύναμο με χρήση 2 διαδοχικών εντολών, δηλαδή *#pragma omp parallel for* είναι ισοδύναμο με ένα *#pragma omp parallel* που ακολουθείται από ένα *#pragma omp for*.

- … Πολλές Ακόμη (Δείτε το Manual του OpenMP).

Βιβλιογραφία

- *Parallel Programming*, B. Wilkinson, M. Allen, Prentice Hall, 2nd Ed. 2005.
- *Introduction to Parallel Computing*, A. Grama, G. Karypis, V. Kumar, A. Gupta, Addison-Wesley, 2003.
- *Parallel Computing: Theory and Practice*, M. J. Quinn, McGraw-Hill, 1994.
- POSIX Threads: http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html
- OpenMP: http://openmp.org/wp/

Τέλος Ενότητας