Παράλληλη Επεξεργασία

Φροντιστήριο:

- *Διαμοίραση έργου και συγχρονισμός στο OpenMP*

Εργαστήριο Πληροφοριακών Συστημάτων Υψηλής Επίδοσης

Parallel and Distributed Systems Group

Παράλληλες Περιοχές

- Καθορίζονται από την οδηγία parallel
- Είναι η πιο βασική οδηγία στο OpenMP
- Όταν ένα νήμα (master thread) συναντήσει την οδηγία parallel δημιουργεί επιπλέον νήματα που θα εκτελεστούν παράλληλα.
- Κάθε νήμα συμπεριλαμβανομένου και του master εκτελεί για λογαριασμό του τον κώδικα που περιλαμβάνει η οδηγία parallel
- Στο τέλος της οδηγίας υπονοείται από την υλοποίηση ένα barrier πέραν του οποίου συνεχίζει την εκτέλεση του μόνο το master thread

Hello World!

```
#include <stdio.h>
#include <omp.h>

int main(int argc, char **argv)
{
    int id;

#pragma omp parallel private(id)
      {
          id = omp_get_thread_num();
          printf("Hello World from %d\n", id);
      }

   fprintf(stderr, "Program terminated\n");
   return 0;
}
```

Διαμοίραση Έργου

- Οι οδηγίες διαμοίρασης έργου (work sharing directives) τοποθετούνται στα πλαίσια μιας οδηγίας parallel για να αναθέσουν κομμάτια της δουλειάς στα νήματα που έχουν ήδη δημιουργηθεί
- Οι οδηγίες αυτές δεν δημιουργούν επιπλέον νήματα
- Δεν υπονοούν barrier στο σημείο εισόδου
- Υπονοούν barrier στο σημείο εξόδου (όμως με δυνατότητα επιλογής στη χρήση του)

Τύποι Οδηγιών Διαμοίρασης Έργου

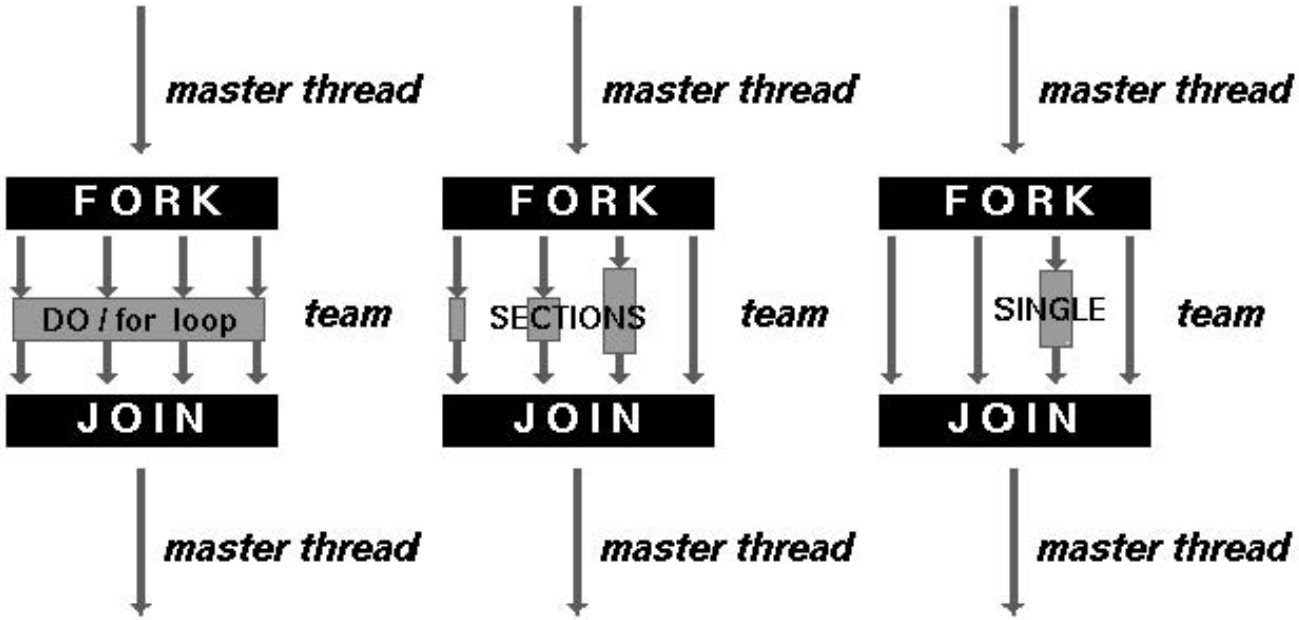
Οι οδηγίες διαμοίρασης έργου είναι 3:

- For
  Προκαλεί την διαμοίραση των επαναλήψεων ενός loop ανάμεσα στα νήματα της παράλληλης περιοχής (υλοποιεί data parallelism)

- Sections
  Επιτρέπει τη ρητή διαμοίραση έργου ξεχωριστών τμημάτων κώδικα (υλοποιεί functional parallelism)

- Single
  Υποδεικνύει την ακολουθιακή εκτέλεση ενός τμήματος κώδικα από ένα νήμα

Τύποι Οδηγιών Διαμοίρασης Έργου



Περιορισμοί:

- Οι οδηγίες πρέπει να συναντώνται είτε από όλα τα νήματα είτε από κανένα
- Πρέπει να συναντώνται με την ίδια σειρά από όλα
- Πρέπει να περιλαμβάνονται σε μια οδηγία **parallel**

Οδηγία for

Ακολουθιακό

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

Παράλληλη περιοχή

```
#pragma omp parallel
{   int id, i, tnum, istart, iend;
    id = omp_get_thread_num();
    tnum = omp_get_num_threads();

    istart = id * N / tnum;
    iend = (id+i) * N / tnum ;
    for(i=istart; i<iend; i++)  { a[i] = a[i] + b[i];}
}
```

Παράλληλη περιοχή + διαμοίραση έργου μέσω for

```
#pragma omp parallel
{
    #pragma omp for schedule(static)
        for(i=0; i<N; i++)  { a[i] = a[i] + b[i];}
}
```

Πρόταση schedule

- Υποδεικνύει πως ανατίθενται οι επαναλήψεις του loop στα νήματα. Επιλογές:
- schedule (static [,chunk])
  Οι επαναλήψεις μοιράζονται εξ αρχής σε τμήματα μεγέθους N/chunk ή N/#νημάτων αν δεν δίνεται μέγεθος
- schedule (dynamic [,chunk])
  Τα τμήματα ανατίθενται δυναμικά κατά την εκτέλεση. Default chunk = 1
- schedule (guided [,chunk])
  Δυναμική ανάθεση τμημάτων εκθετικά μειούμενου μεγέθους. Μέγεθος πρώτου N/#νημάτων. Το chunk συμβολίζει το ελάχιστο μέγεθος ενός τμήματος. Default chunk = 1
- schedule (runtime)
  Η επιλογή της πολιτικής γίνεται κατά το χρόνο εκτέλεσης, βάσει της τιμής της μεταβλητής περιβάλλοντος OMP_SCHEDULE

Οδηγία for

Περιορισμοί

- Το for loop πρέπει να είναι καθορισμένου μεγέθους
- Δεν πρέπει να χρησιμοποιούνται εντολές εξόδου από το loop (branch instructions)
- Το μέγεθος των chunks δεν πρέπει να καθορίζεται δυναμικά από το index του loop

Οδηγία sections

```
#pragma omp parallel
{
    #pragma omp sections
    {
        #pragma omp section
            x_calculation();
        #pragma omp section
            y_calculation();
        #pragma omp section
            z_calculation();
    }
}
```

- Η διαμοίραση έργου με την οδηγία γίνεται ρητά. Σε κάθε νήμα ανατίθεται ένα section. Τα sections εκτελούνται ανεξάρτητα, ενώ στο τέλος της οδηγίας υπονοείται ένα barrier το οποίο μπορεί να παρακαμφτεί με χρήση της πρότασης nowait

Οδηγία single

```
#pragma omp parallel
{
    #pragma omp sections
    {
        #pragma omp section
            x_calculation();
        #pragma omp section
            y_calculation();
    }
    #pragma omp single
        z_calculation();
}
```

- Το τμήμα κώδικα που περιλαμβάνεται στην οδηγία single εκτελείται μόνο από ένα νήμα, ενώ τα υπόλοιπα περιμένουν στο barrier που υπονοείται στο τέλος της οδηγίας (εκτός αν ορίζεται διαφορετικά μέσω της πρότασης nowait)

Συνδυασμός Οδηγιών Διαμοίρασης Έργου και Παράλληλης Περιοχής

- Λόγω της συχνής χρήσης των οδηγιών for και sections οι οδηγίες συνδυάζονται με την οδηγία **parallel** σε μία οδηγία αντίστοιχα

```
int saxpy(int a, int y, double[] z, double[] x, int N){
#pragma omp parallel for schedule(guided)
    for(i = 0; i < N; i++){
        z[i] = a * x[i] + y
}
```

- Με παρόμοιο τρόπο συνδυάζεται και η οδηγία sections στην οδηγία **parallel sections**

Πρόταση reduction

- Συντάσσεται ως:
  - reduction (operator : varlist)
- Εφαρμόζεται σε τελεστές για τους οποίους ισχύει η προσεταιριστική και η αντιμεταθετική ιδιότητα.
- Συγκεκριμένα εφαρμόζεται στους: +, *, – και τους λογικούς &, |, ^, &&, || με τις αντίστοιχες αρχικές τιμές.
- Οι μεταβλητές στη λίστα πρέπει να είναι βαθμωτές (σε αντίθεση με τους πίνακες ή τις δομές είναι μεταβλητές που αποθηκεύουν μια τιμή κάθε στιγμή). Επίσης πρέπει να είναι κοινά διαμοιραζόμενες (δηλωμένες shared) στην παράλληλη περιοχή.
- Μέσα στην παράλληλη περιοχή
  - Δημιουργείται ένα τοπικό αντίγραφο και αρχικοποιείται (πχ για + σε 0)
  - Ο τελεστής εφαρμόζεται στο τοπικό αντίγραφο
  - Το αποτέλεσμα συνοψίζεται στην κοινή μεταβλητή στο τέλος της οδηγίας

Παράδειγμα reduction

```
#include <stdio.h>
#include <omp.h>

int main(int argc, char** argv)
{
     int N = atoi(argv[1]);
     int sum = 0;
     int * A = malloc(N * sizeof(int));

     my_init(A);

       #pragma omp parallel for reduction(+ : sum) schedule(static)
          for (i=0; i < N; i++) {
             sum += A[i];
          }

          printf("Total sum = %d\n", sum);
}
```

Συγχρονισμός

- Οι οδηγίες που δίνουν την δυνατότητα συγχρονισμού στο OpenMP είναι οι εξής:
  - MASTER
  - CRITICAL
  - ATOMIC
  - BARRIER
  - FLUSH
  - ORDERED

Οδηγία master

- Ορίζει ότι η περιεχόμενη περιοχή κώδικα θα εκτελεστεί μόνο από το master thread της ομάδας. Τα υπόλοιπα μέλη της ομάδας παρακάμπτουν τον κώδικα

- Δεν υπάρχει υπονοούμενο barrier στο τέλος της οδηγίας

```
#pragma omp master
    structured block
```

Οδηγία critical

- Υλοποιεί τον πιο απλό και εύκολο στη χρήση μηχανισμό αμοιβαίου αποκλεισμού. Κάθε στιγμή μόνο ένα νήμα μπορεί να εκτελεί τον κώδικα που περιέχεται στην οδηγία.

```
int sum = 0;
int psum = 0

#pragma omp parallel private(psum) shared(sum)
{
      psum = 0;
      #pragma omp for
         for (i=0; i < N; i++)
             psum += a[i];

      #pragma omp critical
      {
         sum += psum;
         printf("[%d] partial sum = %d\n", omp_get_thread_num(), sum);
      }
}
```

- Παίρνει προαιρετικά ένα όνομα που διαφοροποιεί τα ξεχωριστά σημεία αμοιβαίου αποκλεισμού στο πρόγραμμα

Οδηγία atomic

- Εφαρμόζεται σε βαθμωτές μεταβλητές και απλές πράξεις.

```
int sum = 0;
int psum = 0

#pragma omp parallel private(psum) shared(sum)
{
      psum = 0;
      #pragma omp for
         for (i=0; i < N; i++)
              psum += a[i];

      #pragma omp atomic
         sum += psum;
}
```

- Αξιοποιεί μηχανισμούς αμοιβαίου αποκλεισμού που βρίσκονται διαθέσιμοι σε επίπεδο υλικού.

Οδηγία barrier

- Η οδηγία barrier συγχρονίζει όλα τα νήματα μιας ομάδας
- Όταν ένα νήμα φτάσει σε μια οδηγία barrier περιμένει όλα τα άλλα νήματα της ομάδας να φτάσουν και αυτά στο barrier. Όταν συμβεί αυτό, ξεκινούν και πάλι όλα μαζί

```
#pragma omp parallel
{
    while (end == false) {
        compare();
        #pragma omp barrier
        exchange();
        #pragma omp barrier
    }
}
```

Οδηγία flush

- Καθορίζει ένα σημείο συγχρονισμού στο οποίο η υλοποίηση πρέπει να εγγυάται μια συνεπή προς όλα τα νήματα της ομάδας εικόνα της μνήμης.
- Όλες οι ορατές στα νήματα μεταβλητές γράφονται στη μνήμη σε αυτό το σημείο
- Προαιρετικά μπορεί να δεχθεί μια λίστα από μεταβλητές-ορίσματα που θα ανανεώσει ώστε να μην χρειαστεί να ανανεωθούν όλες
- Η οδηγία flush υπονοείται επίσης στις παρακάτω οδηγίες
  - barrier
  - critical- τόσο στην είσοδο όσο και στην έξοδο
  - ordered- τόσο στην είσοδο όσο και στην έξοδο
  - parallel- στην έξοδο
  - for- στην έξοδο
  - sections- στην έξοδο
  - single- στην έξοδο

Οδηγία ordered

- Καθορίζει ότι οι επαναλήψεις του περιεχόμενου loop θα εκτελεστούν με τη σειρά που θα εκτελούνταν σε ένα μονοεπεξεργαστικό σύστημα
- Μπορεί να εμφανιστεί μόνο στο περιεχόμενο των οδηγιών *for* | *parallel for*
- Σε κάθε χρονική στιγμή μόνο ένα νήμα μπορεί να βρίσκεται στο τμήμα που περικλείει η *ordered*

Συναρτήσεις Βιβλιοθήκης Χρόνου Εκτέλεσης

- Συναρτήσεις ελέγχου/αλλαγής του αριθμού των νημάτων
  - omp_get_num_threads(), omp_set_num_threads, omp_get_thread_num()
- Ενεργοποίηση/απενεργοποίηση της εμφώλευσης και της δυναμικής εκτέλεσης
  - omp_set_nested(), omp_set_dynamic(), omp_get_nested(), omp_get_dynamic()
- Ελεγχος για την εκτέλεση σε παράλληλη περιοχή
  - omp_in_parallel()
- Ελεγχος του αριθμού των επεξεργαστών
  - omp_num_procs
- Συναρτήσεις αμοιβαίου αποκλεισμού
  - omp_init_lock(), omp_set_lock(), omp_unset_lock(), omp_test_lock()

Μεταβλητές Περιβάλλοντος

- Καθορισμός της πολιτικής ανάθεσης επαναλήψεων
  - OMP_SCHEDULE “schedule[, chunk_size]”
- Προκαθορισμένος αριθμός νημάτων
  - OMP_NUM_THREADS *int_literal*
- Ενεργοποίηση δυναμικής εκτέλεσης
  - OMP_DYNAMIC TRUE || FALSE
- Ενεργοποίηση εμφωλευμένου παραλληλισμού
  - OMP_NESTED TRUE || FALSE

Μετάφραση Προγραμμάτων

- Σε περιβάλλον όπου παρέχεται ο κατάλληλος compiler (π.χ. GNU C compiler > 4.2 version) συνήθως απαιτούνται:

  - Να συμπεριληφθεί το header file <omp.h>
  - Να συμπεριληφθεί στον compiler η σήμανση -fopenmp
    πχ shell> gcc -fopenmp -o executable source.c

- *Απορίες και συζήτηση μέσω του forum του μαθήματος στο My.CEID*
- *e-mail επικοινωνίας kik@hpclab.ceid.upatras.gr*
- *Ανακοίνωση ασκήσεων και υλικό στην ιστοσελίδα του μαθήματος http://parallel.hpclab.ceid.upatras.gr/*