ΕΡΓΑΛΕΙΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΑΡΑΛΛΗΛΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΜΕ OpenMP (2ο Μέρος)

Νίκος Τρυφωνίδης

Εφαρμογή 7: Ανισορροπία

- Το πρόγραμμα *imbalance.c* περιέχει ένα loop το οποίο έχει μεγαλύτερη εργασία σε ορισμένα iterations.

1. Τρέξτε το πρόγραμμα σειριακά και σημειώστε το χρόνο εκτέλεσης και το αποτέλεσμα.
2. Δοκιμάστε να παραλληλοποιήσετε το πρόγραμμα με απλό διαχωρισμό του loop (static schedule). Τί παρατηρείτε;

Εφαρμογή 7: Ανισορροπία

3. Δοκιμάστε το dynamic schedule, με διάφορα chunk sizes.

4. Αφού δείτε τον κώδικα, δοκιμάστε αν μπορείτε να έχετε καλή απόδοση με static schedule, χρησιμοποιώντας διάφορα chunk sizes.

Μέρος 7ο:

ΕΠΙΠΛΕΟΝ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Επιπλέον;

- Όσα αναφέρθηκαν εώς τώρα είναι η **βάση** του παράλληλου προγραμματισμού με OpenMP.
- Μπορούν όμως να εφαρμοστούν σε μεγάλη ποικιλία υπολογιστικών προγραμμάτων.
- Στη συνέχεια θα αναφέρουμε ορισμένα επιπλέον χαρακτηριστικά που μπορεί να φανούν χρήσιμα σε κάποιες περιπτώσεις.

Single

- **Single Directive:** Ο κώδικας που εσωκλείεται εκτελείται *μόνο από ένα thread*.

```
#pragma omp parallel
{
   (parallel code)
   #pragma omp single
   {
      (“serial” code)
   }
   (parallel code)
}
```

Μόνο ένα thread θα εκτελέσει τον κώδικα μέσα στο single region!

Τα υπόλοιπα περιμένουν στο τέλος.

Master

- **Master Directive:** Ο κώδικας που εσωκλείεται εκτελείται *μόνο από το Master thread (0).*

```
#pragma omp parallel
{
   (parallel code)
   #pragma omp master
   {
      (“serial” code)
   }
   (parallel code)
}
```

Μόνο το master thread θα εκτελέσει τον κώδικα μέσα στο master region!

ΠΡΟΣΟΧΗ:Τα υπόλοιπα συνεχίζουν – ΔΕΝ περιμένουν στο τέλος!

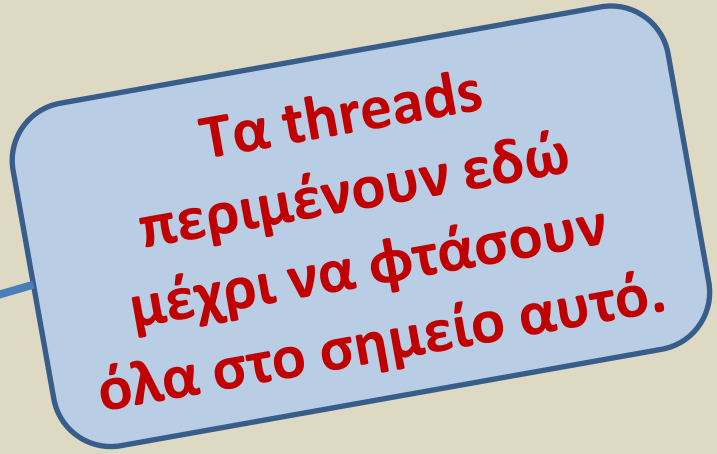
Barrier

- Τα barriers είναι ο βασικός τρόπος συγχρονισμού.
- Όταν ένα thread συναντήσει ένα barrier, περιμένει σε εκείνο το σημείο μέχρι να φτάσουν όλα τα threads εκεί.
- Μόλις γίνει αυτό, τα threads συνεχίζουν την εκτέλεση.

Barrier

```
#pragma omp parallel
{
  (parallel code)
    .
    .
    .
    #pragma omp barrier

 (parallel code)
    .
    .
    .
}
```



Barrier

- **Υπενθύμιση**: Υπάρχει νοητό (implicit) barrier στο τέλος των parallel regions, των worksharing constructs, των single regions κτλ.

- Όταν δε γνωρίζουμε αν ένα construct που χρησιμοποιούμε περιλαμβάνει νοητό barrier, πρέπει να το ελέγχουμε.

Barrier

- Σε γενικές γραμμές, τα νοητά barriers είναι τοποθετημένα ώστε να καλύπτουν τις ανάγκες του προγράμματος.

- Αν δούμε ότι χρησιμοποιούμε πολλά barriers, είναι πιθανό να είναι περιττά.

Άχρηστα Νοητά Barriers

- Σε ορισμένες περιπτώσεις, τα νοητά barriers μπορεί να είναι περιττά, και απλά να επιβαρύνουν το πρόγραμμα με άχρηστες αναμονές.

- Τότε, μπορούμε να χρησιμοποιούμε το *nowait clause.*

Άχρηστα Νοητά Barriers

```
#pragma omp parallel (clauses etc)
{
  #pragma omp for nowait
  for (i=0; i<N; i++) {
    (1st loop code)
  }

  #pragma omp for
  for (i=0; i<N; i++) {
    (2nd loop code)
  }
}
```

Τα threads **δεν** περιμένουν εδώ, συνεχίζουν κατευθείαν στο 2ο loop.

ΠΡΟΣΟΧΗ: Σωστό μόνο αν το 2ο loop δεν εξαρτάται από υπολογισμούς που έγιναν στο 1ο.

Μέρος 8ο:

OPENMP TASKS

OpenMP Tasks

- Τα Tasks είναι σχετικά νέα προσθήκη στο OpenMP (έκδοση 3.0 - 2008).
- Task:
  - Κομμάτι κώδικα και αντίστοιχης μνήμης εντός μιας parallel region.
  - “Πακετάρεται” και περιμένει εκτέλεση από κάποιο thread.
  - **Κάποιο** thread το εκτελεί, **κάποια** στιγμή (!!).

OpenMP Tasks

- Τα tasks ουσιαστικά επιτρέπουν την παραλληλοποίηση μιας ευρύτερης ποικιλίας μορφών επανάληψης
- Χαρακτηριστικό παράδειγμα: while loops χωρίς γνωστό αριθμό επαναλήψεων κατά την εκκίνηση

OpenMP Tasks: Δομή

```
#pragma omp parallel (scope clauses)
{
  #pragma omp single
  {
    while (condition) {
      #pragma omp task (task scope clauses)
      (work)
    }
  }
}
```

OpenMP Tasks: Χρήση

- Μια περίπτωση για την οποία τα tasks είναι χρήσιμα είναι η παραλληλοποίηση δομών linked-list.

- Γενικά είναι καλό να αποφεύγεται η χρήση τους όταν υπάρχει εναλλακτική λύση, καθώς η δημιουργία και η διανομή τους έχει μεγάλο κόστος.

OpenMP Tasks: Χρήση

- Είναι προτιμότερο να έχουμε λίγα και βαριά tasks, παρά πολλά ελαφριά.
- Περαιτέρω διερεύνηση των tasks ξεφεύγει από το πλαίσιο μιας εισαγωγής στο OpenMP, αλλά είναι διαθέσιμη στο επιπλέον υλικό που προτείνεται στο τέλος.

Εφαρμογή 8: Poisson

- Γράψτε ένα πρόγραμμα που να επιλύει την μονοδιάστατη εξίσωση Poisson παράλληλα (σε μορφή προβλήματος συνοριακών τιμών), με χρήση OpenMP.

- Αν το πρόγραμμα σταματά με έλεγχο υπολοίπου (residual), δοκιμάστε να παραλληλοποιήσετε και αυτή τη διαδικασία.

Υλικό OpenMP

- http://openmp.org/wp/
- https://computing.llnl.gov/tutorials/openMP/

- Βιβλία:
  - **Using OpenMP** (Barbara Chapman, Gabriele Jost and Ruud van der Pas)
  - **Parallel Programming In OpenMP** (Chandra, Dagum et al)

Ευχαριστώ!

Νίκος Τρυφωνίδης