OpenMP

Exercices pour le Module 5
INF8601 Systèmes informatiques parallèles

Michel Dagenais

École Polytechnique de Montréal
Département de génie informatique et génie logiciel

Une course

Un outil de détection de problème de synchronisation vous indique une course dans la section réalisée par votre co-équipier qui affirme que tout est correct. Qui croire?

```
found = 0;
#pragma omp parallel for
for(i = 0 ; i < n ; i++)
{ if(a[i] == key) found = 1; }
if(found) { ... }
```

Une course

Un outil de détection de problème de synchronisation vous indique une course dans la section réalisée par votre co-équipier qui affirme que tout est correct. Qui croire?

```
found = 0;
#pragma omp parallel for
for(i = 0 ; i < n ; i++)
{ if(a[i] == key) found = 1; }
if(found) { ... }
```

Un seul changement peut arriver, toujours le même, found=1. Il y a effectivement une course mais le résultat final ne dépend pas de la course.

Enlever les barrières

4.2 Votre co-équipier se plaint que votre programme ne profite pas de la performance disponible en raison de la barrière implicite entre les deux boucles. Il veut les mettre nowait ou dans des sections. Est-ce correct? Avez-vous mieux à proposer?

```
#pragma omp parallel
{ #pragma omp for
  for(i = 0; i < n; i++)
  { a[i] = f1(i); }

  #pragma omp for
  for(i = 0; i < n; i++)
  { b[i] = a[i] + f2(i); }
}
```

Enlever les barrières

Avec un ordonnancement statique, nowait serait possible. Mieux encore, on peut fusionner les deux boucles.

```
#pragma omp parallel
{ #pragma omp for
  for(i = 0; i < n; i++)
  { a[i] = f1(i);
    b[i] = a[i] + f2(i);
  }
}
```

Barrières optionnelles

Votre co-équipier a modifié votre programme en ajoutant des nowait et obtient un gain de performance appréciable mais ses résultats seront-ils corrects et fiables?

```
#pragma omp parallel
{ #pragma omp for schedule(static) nowait
  for (i=0; i<n; i++) c[i] = (a[i] + b[i]) / 2.0f;

  #pragma omp for schedule(static) nowait
  for (i=0; i<n; i++) z[i] = sqrtf(c[i]);

  #pragma omp for schedule(static) nowait
  for (i=1; i<=n; i++) y[i] = z[i-1] + a[i];
}
```

Barrières optionnelles

Entre les deux premières boucles, l'ordonnancement statique règle clairement le problème. C'est aussi le cas pour la troisième boucle car tout est décalé de 1 (indice de boucle et accès de z); le même fil opérera sur la même valeur de z. Le dernier nowait est inutile puisque c'est la dernière boucle du parallel.

Travail de longueur variable

Votre programme comporte une boucle dont la longueur de chaque itération varie grandement. Discutez de différents mécanismes qui pourraient être utilisés pour assurer néanmoins une bonne parallélisation.

Travail de longueur variable

Votre programme comporte une boucle dont la longueur de chaque itération varie grandement. Discutez de différents mécanismes qui pourraient être utilisés pour assurer néanmoins une bonne parallélisation.

Avec un ordonnancement de boucle dynamique ou guidé, possiblement avec une granularité assez fine, le nombre d'itérations par fil sera ajusté automatiquement.