Parallélisation automatique avec Intel compiler

Les compilateurs Intel ® C++ et Fortran ont la capacité d'analyser le flux de données dans les boucles d'un programme afin de déterminer si leurs itérations peuvent être exécutées en parallèle. La parallélisation automatique peut réduire le temps d'exécution sur des architectures multi-coeurs. Le compilateur aide le programmeur à :

- Rechercher des boucles qui sont de bons candidats pour une exécution en parallèle
- Effectuer une analyse de flux de données pour vérifier une exécution parallèle

Pour activer la parallélisation automatique, il suffit de rajouter l'option -parallel. Toutefois, le succès de la parallélisation est soumise dépend de certains critères :

1. Le nombre d'itérations doit être connu avant d'entrer dans la boucle.
2. Le corps de la boucle ne doit pas contenir de branchements (goto,...) vers l'intérieur ou l'extérieur de la boucle.
3. Les itérations de la boucle doivent être indépendantes : absence de dépendances entre les itérations

Lorsque le compilateur est incapable de paralléliser automatiquement les boucles complexes, OpenMP est la solution préférée. Dans ce cas, c'est au programmeur de s'assurer que les boucles peuvent être parallélisées sans problème.

Exemple

Prenons un exemple en fortran :

```
PROGRAM TEST
PARAMETER (N=10000000)
REAL A, C(N)

DO I = 1, N
 A = 2 * I - 1
 C(I) = SQRT(A)
ENDDO

PRINT*, N, C(1), C(N)
END
```

L'analyse de flux de données confirme que la boucle ne contient pas de dépendances de données. Le compilateur génère un code qui distribue les itérations sur l'ensemble de coeurs (threads) disponibles à l'exécution.

On peut spécifier le nombre de threads à utiliser à la OpenMP on exportant la variable d'environnement : $OMP_NUM_THREADS.

Par exemple :

```
$ export $OMP_NUM_THREADS=8
```

Utilise 8 threads (coeurs) pour l'exécution.

Compilation :

```
$ ifort -parallel test.f90 -o test

test.f90(4): (col. 3) remark: LOOP WAS AUTO-PARALLELIZED
```

Le compilateur est capable de générer un rapport sur les boucles qui ne pouvait pas paralléliser en utilisant l'option : -par-report3

Voici un exemple :

```
void add (int k, float *a, float *b)
{
  for (int i = 1; i < 10000; i++)
  a[i] = a[i+k] + b[i];
}
```

```
icc -parallel -par-report3 test.c -o test
```

```
    procedure: add
    test.c(7): (col. 1) remark: parallel dependence: assumed ANTI dependence
between a line 7 and a line 7. flow data dependence assumed
    ...
    test.c(7): (col. 1) remark: parallel dependence: assumed FLOW dependence
between a line 7 and b line 7.
```

Analyse

Parce que le compilateur ne connaît pas la valeur de k, il suppose que les itérations dépendent les unes des autres, comme par exemple si k est égal à -1. Cependant, le programmeur connaît mieux son application (par exemple, k toujours supérieure à 10000), et peut dans ce cas aider le compilateur à paralléliser la boucle en lui donnant une indication supplémentaire, en insérant la

clause :

#pragma parallel en entrée de la boucle :

```
    void add (int k, float *a, float *b)
    {
    #pragma parallel
    for (int i = 1; i < 10000; i++)
    a[i] = a[i+k] + b[i];
    }
```

```
$ icc -parallel -par-report3 test.c -o test
```

```
procedure: add
test.c(6): (col. 1) remark: LOOP WAS AUTO-PARALLELIZED.
```

Voir aussi

Parallélisation avec Introduction à OpenMP

Liens

http://software.intel.com/en-us/articles/intel-guide-for-developing-multithreaded-applications/