Examen du cours I3
Programmation hybride et multi-cœurs

Vendredi 15 février 2019 - durée 3 heures
Supports de cours autorisés.

Dans les questions où on demande d'écrire des lignes de code, les erreurs de syntaxe ne seront pas prises en compte (ponctuation, nom exact des fonctions, ordre des arguments, etc.). Du moment que vous indiquez clairement ce que fait chaque ligne de code ajoutée pour répondre aux questions.

Question n° 1 (2 points)

- Définir les localités spatiale et temporelle.
- Des deux versions ci-dessous du code qui calculent le produit matrice-vecteur, laquelle sera probablement plus rapide pour `n` et `m` grands ? Même question pour `n` et `m` petits ? Motivez vos réponses.

```
1 void matmult1(const std::vector<double> & A,
2                const std::vector<double> & V,
3                std::vector<double> & W)
4 {
5   size_t i,j,n = W.size(), m = V.size();
6   for (i=0; i<n; i++)
7     for (j=0; j<m; j++)
8       W[i] += A[i*m + j] * V[j];
9 }
```

```
1 void matmult2(const std::vector<double> & A,
2                const std::vector<double> & V,
3                std::vector<double> & W)
4 {
5   size_t i,j,n = W.size(), m = V.size();
6   for (j=0; j<m; j++)
7     for (i=0; i<n; i++)
8       W[i] += A[i*m + j] * V[j];
9 }
```

Question nº 2 (5 points)

On veut paralléliser avec OpenMP, la fonction C++ :

```
1  void maxlocal(std::vector<double> & v,
2                std::vector<int> & imax, std::size_t &nmax) {
3    std::size_t i, n = v.size();
4    std::size_t smax = imax.size();
5    nmax = 0;
6    for (i=1; i<n-1; i++)
7      if ((v[i-1] < v[i]) && (v[i] > v[i+1])) {
8        imax[nmax] = i;
9        nmax += 1;
10       if (nmax >= smax) break;
11     }
12 }
```

> Le but de cette fonction est de calculer, sur les composantes d'un vecteur `v`, les indices des `maxima locaux` (i tels que $v_i > v_{i-1}$ et $v_i > v_{i+1}$).
> On suppose que le nombre de maxima locaux est inférieur à la taille du vecteur d'entiers `imax` (dans lequel seront rangés les indices de maxima locaux).

La parallélisation de cette fonction en ajoutant une pragma simple suivant le modèle "OpenMP grain fin" est impossible (le compilateur refuse de compiler quand on ajoute la pragma), ou en tout cas compliquée.

- Expliquer pourquoi.
- Modifier la fonction en la parallélisant suivant le modèle "OpenMP grain grossier".
- La version parallèle fournit un résultat correct mais peut-être différent de celui de la version séquentielle. Quelle est cette différence ?
- Question optionnelle : indiquer une méthode pour obtenir si possible un résultat identique.

Question nº 3 (2 points)
Question cours Hybride OpenMP-MPI

Question nº 4 (3 points)
Hybride OpenMP-MPI

Question nº 5 (2 points)

Question cours Cuda

Question n° 6 (1 point)

Décrivez les principales différences entre Cuda et OpenCL (mode de définition d'un noyau de calcul et exécution de ce noyau sur un GPU).

Question n° 7 (5 points)
Programmation Cuda

Total : 20 points