POLYTECH GRENOBLE

UNIVERSITÉ Grenoble Alpes

VT 13/10/2017 - Anthony Geourjon

Introduction

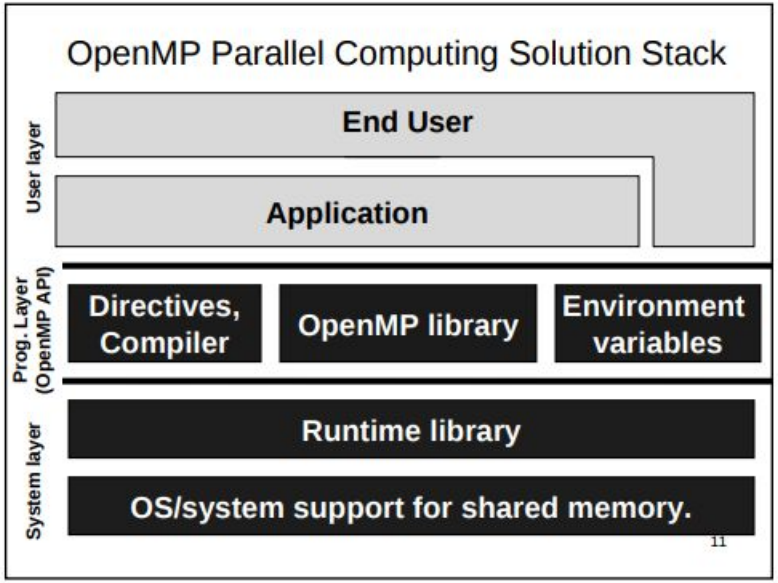
- 1.0 publié en 1997, 4.0 en 2013 (toujours maintenu)
- Développé par les acteurs de l'HPC (privé et recherche)
- Avènement de l'HPC et des machine multi coeurs
- Avant OpenMP développement d'applications parallèles difficile
    - Bibliothèque pthread.h
    - Synchronisation difficile des tâches et programmation difficile.

OpenMP, qu'est ce que c'est ?

- Interface de programmation pour le calcul parallèle sur architecture à mémoire partagée
- Supporté par la majorité des plateformes
- Pour C, C++ et Fortran
- Peu d'alternative à part Intel Threading Building Blocks
- Bibliothèque simple, standard et portable

Fonctionnement

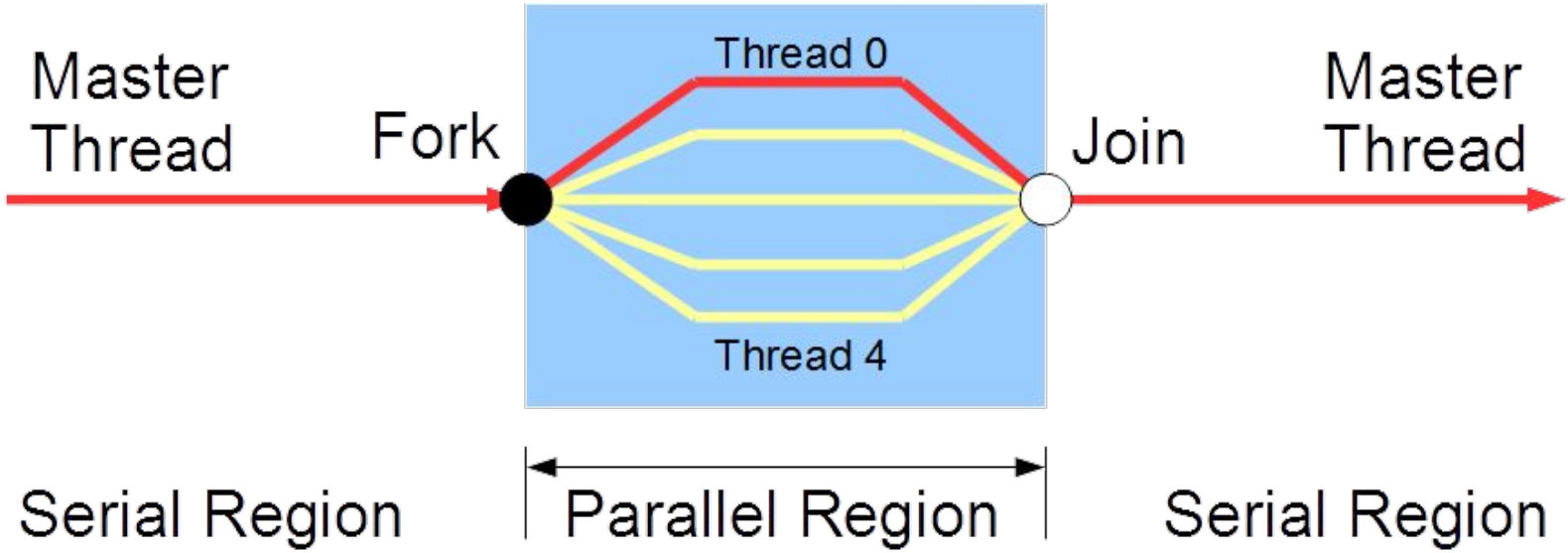
- S’utilise en ajoutant une option de compilation au compilateur
  - gcc -o omp_hello -fopenmp omp_hello.c
-



Fonctionnement

- Paralléliser les boucles grâce à des directives et des variables d'environnement
- 3 types de directives :
  - Partage du travail
  - Partage des données
  - Synchronisation
- Basé sur le modèle fork and join
- OpenMP cible les architectures à mémoire partagée ; les variables sont partagées
  - Gestion de la portée des variables dans les sections parallèles.

Modèle Fork-Join



Exemple de directives

- Création de régions parallèles
  - **parallel** : crée une région parallèle sur le modèle fork-join
- Partage du travail
  - **for** : partage des itérations d'une boucle parallèle
  - **sections** : définit des blocs à exécuter en parallèle
  - **single** : déclare un bloc à exécuter par un seul thread
- Synchronisation
  - **master** : déclare un bloc à exécuter par le thread maître
  - **critical** : bloc à n'exécuter qu'un thread à la fois
  - **atomic** : instruction dont l'écriture mémoire est atomique
  - **barrier** : attente que tous les threads arrivent à ce point

Avantages et Inconvénients

- Pour
  - Les code séquentiels et parallèles sont les mêmes -> maintenance simplifiée
  - On peut paralléliser un code existant sans devoir le changer en profondeur, Si bien utilisé on gagne énormément de temps dans les goulots d'étranglement de programme
  - Bien utilisé on améliore singulièrement les performances d'un code
- Contre
  - L'utilisateur devra s'assurer que les conflits d'accès à la mémoire sont synchronisés
  - Ne favorise pas la scalabilité des accès mémoire