EXERCICES ET TRAVAUX PRATIQUE DU COURS DE CALCUL INTENSIF

KATUBAYEMWO FRIDOLIN

LE 28/03/2023

QUESTION 1: réponse

Un makefile se compose d'une section cible, dépendances et règles. Les dépendances sont les éléments ou le code source nécessaires pour créer une cible ; la cible est le plus souvent un nom d'exécutable ou de fichier objet. Les règles sont les commandes nécessaires pour créer la cible.

QUESTION2:Réponse

Le système du makefile permet de simplifier la compilation. Ce système se résume en un simple fichier Makefile. Ce fichier liste tous les fichiers du projet et permet d'organiser ce dernier en morceaux indépendants. Le fichier Makefile est lu par la commande make, ainsi pour compiler un projet, avec un fichier Makefile qui va bien, il suffit de taper la commande "make" sans aucun arguments. Structure du fichier Makefile : Un fichier Makefile est divisé en structures telles que:
.cible
.dépendances
.commandes

1 QUESTION3

a)Réponse

On distingue classiquement quatre types principaux de parallélisme (Taxonomie de Flynn-Tanenbaum): SISD, SIMD, MISD et MIMD. Cette classification est basée sur les notions de flot de contrôle (deux premières lettres, I voulant dire "Instruction") et flot de données (deux dernières lettres, D voulant dire Data).

1.1 b)Réponse

Flot d'instructions x Flot de données:
.Single Instruction Single Data (SISD)
.Single Instruction Multiple Data (SIMD)
.Multiple Instruction Single Data
.Multiple Instruction Multiple Data (MIMD)
Tout est MIMD

c)Réponse

Le standard OpenMP est clairement dédié aux parallélismes légers. Pour celles et ceux qui ont déjà fait des programmes parallèles (e.g., avec pthreads en C/C++), vous allez constater que l'interface OpenMP simplifie grandement les mécanismes de parallélisation du code. Pour ceux qui n'ont jamais fait de programmes parallèles, vous allez découvrir que vous pouvez probablement paralléliser une partie de vos programmes avec très peu d'effort.
Il est à noter que les implémentations d'OpenMP existent pour les langages C/C++ et Fortran. Dans la suite de cet article, je n'évoquerai que l'API C/C++ d'OpenMP.
tandis que pour openMPI

Il existe plusieurs implémentations du standard MPI. Comme vous l'aurez compris, Open MPI est une implémentation libre et open source de ce protocole. Il en résulte qu'en théorie, je parlerai tout autant de MPI que d'Open MPI (et que de toute autre implémentation) dans la suite.
Au sein d'un groupe de communication, certaines fonctions de MPI nécessitent que tous les processus du groupe communiquent ensemble. Ces routines sont dites collectives. A contrario, certaines fonctions peuvent s'exécuter indépendamment de tout ou partie des autres processus. Ces routines sont alors dites non collectives.
Par défaut, MPI définit un groupe de communication universel regroupant tous les processus. Il est possible de créer d'autres communicateurs et ainsi de définir des sous-groupes de processus. C'est pourquoi MPI distingue les communications dites intracommunicantes (au sein d'un même groupe) et des communications dites intercommunicantes (entre des groupes distincts).