Série d'exercices #3

IFT-2245

29 janvier 2019

3.1 Amdahl

Donner la loi d'Amdahl et comment faire pour retrouver la formule.

1. À quoi cette équation s'applique.
2. Les conséquences des valeurs extrêmes valides.

3.2 Concurrence vs parallélisme

Est-il possible d'avoir de la concurrence sans parallélisme ?
Donnez une définition de ces deux concepts.

3.3 Création de *thread*

Quand un *thread* crée un nouveau *thread*, quelles parties, parmis les suivantes, sont-elles partagées entre les deux *threads* :

— Les registres
— Le tas
— Les variables globales
— La pile

3.4 L'état d'un *thread*

Pour chaque flèche dans le diagramme ci-bas, décrire une condition qui causerait un fil (thread) de passer d'un état à l'autre. Indiquer "impossible" si cela ne peut pas se produire. S'il y a lieu, indiquer aussi quelle action d'un *context switch* s'y produit (save ou restore).

En exécution	$\overset{2}{\Rightarrow}$	Prêt	$\overset{4}{\Rightarrow}$	Bloqué	$\overset{6}{\Rightarrow}$	En exécution
(*running*)	$\overset{\lceil}{1}]\Leftarrow$	(*ready*)	$\overset{\lceil}{3}]\Leftarrow$	(*blocked*)	$\overset{\lceil}{5}]\Leftarrow$	(*running*)

3.5 User threads

Est-il possible d'améliorer la performance d'une application multi-threaded utilisant des user thread en exécutant le programme sur une machine ayant plusieurs processeurs plutôt que sur une machine ayant un seul processeur. Pourquoi ?

3.6 Multithreading

Considérer un programme qui ouvre un fichier, effectue des opérations qui sont *cpu-bound*, puis écrit les résultats à l'intérieur d'un fichier.

Ce programme s'exécute sur une machine ayant 2 processeurs double cœur. Discuter comment vous pourriez améliorer la performance avec du multithreading.

3.7 Fork & friends

Soit le code suivant.

```
pid_t pid1 = fork();
if (pid1 == 0)
{
  pid_t pid2 = fork();
  thread_create(...);
  thread_join(...)

  if (pid2 > 0)
    wait(NULL);
}
pid_t pid3 = fork();
```

1. Combien de processus sont créés ?
2. Combien de threads sont créés ?

Donner toutes les suppositions que vous avez faites et considérer un autre ensemble de suppositions.

3.8 Ordonnancement de threads

Dans le cours ont a vu que les processus sont ordonnancés afin de bien répartir le temps de calcul entre eux. Suite à l'introduction des threads, quels nouveaux facteurs et nouvelles problématiques sont à considérer pour l'ordonnancement du modèle de processus permettant des threads ? (Considérez les unités à ordonnancer, les informations partagées, l'impact sur la performance, etc.)

3.9 Questions complémentaires

— Pourquoi voudrait-on utiliser des threads plutôt que des processus et inversement ?

— Dans quelles circonstances est-il préférable d'avoir une solution multithreaded qu'une solution single-threaded sur un système n'ayant qu'un seul processeur.

— Utiliser l'exemple d'une fabrique de chaussures pour expliquer le *data parallelism* (parallélisme de donnée) et *task parallelism* (parallélisme de tâche). [Expliquer en même temps la différence].
De quelle manière la concurrence apparait dans ce problème ?
— Identifier les avantages et les inconvénients de chacun des modéles de multithreading. Lesquels parmis ceux-ci permettent la concurrence. Lesquels permettent du parallélisme.