Labo 1 - Programmation concurrente

Owen Gombas
David Darmanger
28.03.2022
ISC2id-a
HE-ARC

Introduction

Dans ce laboratoire, nous avons dû réaliser une synchronisation des threads afin d'obtenir un résultat cohérent
avec une variable partagée et incrémenté depuis plusieurs threads en même temps.

```
i <- 0

thread 1:
    faire N fois:
        i <- i + 1

thread 2:
    faire N fois:
        i <- i + 1

thread 3:
    faire N fois:
        i <- i + 1

...

thread M:
    faire N fois:
        i <- i + 1

Attend la fin de tous les threads

Afficher i qui devrait en théorie valoir M * N (pas toujours le cas)
```

Sans blockage

Comme expliqué dans le cours, une opération d'incrémentation est divisée en 3 parties et un thread peut interrompre l'incrémentation effectuée par un autre thread ce qui cause une corruption de la valeur de `i` qui ne sera pas celle attendue. L'exécution est rapide.

```
Number of threads:10000
Number of iterations:10000
```

```
Mode:
1) Without locking system
2) With busy waiting
3) With sched_yield(void)
4) With pthread_mutex_t
1
With no blocking system:
We got: 99750432
We wanted: 100000000
Ratio: 99.750427%
```

Avec un système d'attente active

Ce système ne produit également pas de résultats convaincants, c'est le pire. L'exécution reste rapide.

```
Number of threads:10000
Number of iterations:10000
Mode:
1) Without locking system
2) With busy waiting
3) With sched_yield(void)
4) With pthread_mutex_t
2
With busy waiting:
We got: 63790260
We wanted: 100000000
Ratio: 63.790260%
```

Avec `sched_yield`

Ce système ne produit non plus pas un résultat exact à 100%. L'exécution est également rapide.

```
Number of threads:10000
Number of iterations:10000
Mode:
1) Without locking system
2) With busy waiting
3) With sched_yield(void)
4) With pthread_mutex_t
3
With yield:
We got: 99917432
We wanted: 100000000
Ratio: 99.917427%
```

Avec `pthread_mutex_t`

Ce système produit des résultats exacts à 100% cependant l'exécution est plus lente que les autres systèmes bloquants.

```
Number of threads:10000
Number of iterations:10000
Mode:
1) Without locking system
2) With busy waiting
3) With sched_yield(void)
4) With pthread_mutex_t
4
With mutex:
We got: 100000000
We wanted: 100000000
Ratio: 100.000000%
```