Programmation Concurrente

TD – Quelle est la synchronisation la plus efficace

Le compteur partagé (en Java)

On souhaite implémenté un compteur partagé selon différentes techniques afin de mesurer les divers temps d'exécution.

Etape 0. Construisez une classe abstraite `Compteur` proposant la méthode `long incr()`.

Etape 1. Construisez une classe `Compteur` implémentant une méthode `long inc()`. Cette méthode incrémente une variable d'état `compteur` initialisée à 0 lors de la création puis renvoie la valeur du compteur.

Etape 2. Construisez une classe `Compteur_synchronized` qui implémente de manière correcte l'incrémentation du compteur à l'aide du mot clé `synchronized`.

Etape 3. Construisez une classe `Compteur_lock` qui implémente de manière correcte l'incrémentation du compteur à l'aide d'un verrou réentrant.

Etape 4. Construisez une classe `Compteur_rw_lock` qui implémente de manière correcte l'incrémentation du compteur à l'aide d'un verrou read_write.

Etape 5. Construisez une classe `Compteur_semaphore` qui implémente de manière correcte l'incrémentation du compteur à l'aide d'un sémaphore.

Etape 6. Construisez une classe `Compteur_Atomic_Int` qui implémente de manière correcte l'incrémentation du compteur à l'aide d'un `AtomicInteger`.

Vous pouvez aussi ajouter :

- **Utilisation du CAS disponible dans Java**
- **Un algorithme par attente active de type Perterson / Dekker**

Etape 7. Synthèse 1

- Si N est le nombre de coeur de votre station, construisez un processus Main qui crée N threads. Chaque thread exécute 100.000 fois la méthode `inc()`
- Mesurez le temps d'éxécution pour chacune des classes compteurs

- Calculer le temps moyen d'exécution d'une incrémentation selon les différentes classes compteurs

Etape 8. Synthèse 2

- Reproduisez l'expérience ci-dessus mais avec 100*N threads

Etape 9. Synthèse

- Analyser les résultats obtenus. Il peut être intéressant de présenter les résultats sur un graphique.