TP2 – Programmation parallèle : Design Pattern

Exercice 1 : Thread pool (approximation de e-1)

Une approximation de e−1 nous est donnée par la formule suivante :

$$e^{-1} = \sum_{i=0}^{n} \frac{(-1)^i}{i!}$$

Écrire un programme qui calcule une approximation de e−1 en additionnant n termes. Ce programme devra répartir la charge de calculs sur k threads en utilisant le design pattern : thread pool.

Exercice 2 : Pipeline (Natural Language Processing)

En traitement automatique du langage naturel (Natural Language Processing – NLP) l'étape de normalisation d'un texte consiste à convertir un texte dans une forme canonique. Le but de cette normalisation permet de standardiser l'entrée d'un texte pour les futurs traitements.

Par exemple un texte : « INFOS: le tournage du film les 101 dalmatiens a été fait à Avignon☺ » deviendra « infos le tournage du film les cent un dalmatiens a été fait à avignon ☺ ».

Les étapes de normalisation consistent à :

- Passer le texte en minuscule
- Tokeniser le texte (ie. séparer chaque token par des espaces, par exemple : « L'artiste à réaliser une belle peinture. » deviendra [« l' », « artiste », « à », « réaliser », « une », « belle », « peinture »])
- Supprimer la ponctuation
- Transformer les nombres en lettre (ie. « 200 » deviendra « deux cent »).

Réaliser les étapes de normalisation en utilisant le design pattern pipeline, ie. chaque thread sera une étape de normalisation.

Exercice 3 : Loop parallelism (déchiffrage distribué)

Une fonction de hachage cryptographique a pour but de calculer une valeur de hachage d'un mot ou d'une donnée. Le calcul de la valeur de hachage doit être rapide mais retrouver la donnée initiale à partir de la valeur de hachage doit être très difficile. La façon la plus simple de retrouver le mot initial à partir de la valeur de hachage est ce qu'on appelle une recherche force brute. Nous allons appliquer la fonction de hachage sur toutes les combinaisons de caractères possibles jusqu'à arriver à la valeur de hachage que l'on cherche à déchiffrer.

string secret = "3ed7dceaf266cafef032b9d5db224717";

Voici un mot de 5 lettres chiffre en md5. En utilisant le design pattern loop parallelsim, retrouver la chaîne de caractère dont la valeur de hachage correspond à secret.

Vous pouvez utiliser cette page, pour convertir une chaîne de caractère en MD5 : http://www.zedwood.com/article/cpp-md5-function