Structures et algorithmes concurrents

Lab 5 : Single instruction, Multiple Data (SIMD)

Github: https://github.com/furtiveJack/Collections_Concurrente

Vector API (jdk 15)

- API permettant de représenter des vecteurs de données :
 - O Ils sont spécifiques à un type primitif (Short, Int, Float etc..).
 - O Ils représentent un ensemble de plusieurs valeurs de même type.
 - On appelle lanes les valeurs contenues dans le vecteur.
 - O Le nombre de lanes dépend du processeur

VectorSpecies

- La classe VectorSpecies<> représente :
 - Le type du vecteur: SPECIES.elementType.
 - Le nombre de lanes : vectorSpecies.length().
 - O Exemple de déclaration :

private static final VectorSpecies<Integer> SPECIES = IntVector.SPECIES_PREFERRED;

Méthodes de l'API Vector (jdk 15)

- IntVector.fromArray(SPECIES, array, index):
 - o extrait SPECIES.length() valeurs dans le tableau array en commençant à l'indice index et les stocke dans le vecteur renvoyé;
- vector.intoArray(array, i): opération inverse, prend les valeurs du vecteur et les stocke dans le tableau array;
- IntVector.zero(): initialise un nouveau vecteur avec la valeur zéro;
- vector.broadcast(int value): initiliase un vecteur avec la valeur fournie;

Opérations lanewises

- Une opération lanewise est une opération qui s'applique sur chaque paire de lane de deux vecteurs.
 - O Les possibilités sont : add, sub, mul, div, min, max, etc...
- O Les vecteurs doivent contenir le même nombre de lanes!
- La méthode v.reduceLanes (associativeOp) permet de prendre toutes les lanes d'un vecteur et de leur appliquer la même opération.
 - Les opérations sont définies dans la classe VectorOperators.

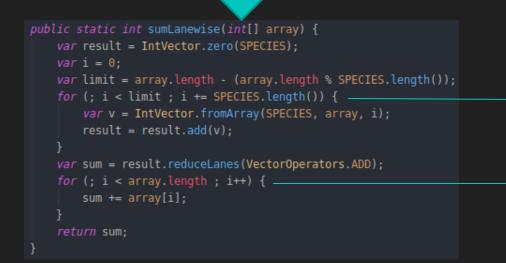
Comparaison Vector API / Fork Join API

- L'API Fork/Join permet d'effectuer un calcul en le combinant sur plusieurs cœurs, ce qui permet de gagner en efficacité (pour les gros calculs)
- L'API des Vector permet de faire plusieurs calculs sur un seul cœur.
- O Il est donc (théoriquement) possible de combiner les deux APIs pour gagner en efficacité.

Exemple d'implémentation

Somme de toutes les valeurs d'un tableau :

```
public static int sumLoop(int[] array) {
    var sum = 0;
    for (var value : array) {
        sum += value;
    }
    return sum;
}
```



On parcourt le tableau en avançant de SPECIES.length() à chaque itération

On finit le parcourt à la main étant donné donné que le nombre d'éléments restants dans le tableau est inférieur à SPECIES.length()