Sorbonne Université

LIP6

Licence Informatique

Stencils 1D – transformations de boucles

Le but de ce TD est d'appliquer les principales transformations de boucles à des stencils.
Notations :

— notation "algo" x, y, n, accès à une case $x(n)$
— notation "C" minuscules pour les variables `x`, `y`, `n`
majuscule pour les tableaux `X`, `Y`, accès à une case `X[i]`

Stencil sans bord

Soit le stencil S_3 qui est la somme de 3 points consécutifs (eq. 1) :

$$y(n) = x(n-1) + x(n) + x(n+1) \tag{1}$$

En appliquant le stencil S_3 à un tableau X de n cases, 3 cas apparaissent (fig. 1) : un cas général et deux cas particuliers, pour chaque bord du tableau. Pour produire la première case `Y[0]`, la case `X[-1]` est nécessaire mais n'existe pas et pour produire la dernière case `Y[n-1]`, la case `X[(n-1)+1]` est nécessaire mais n'existe pas non plus.

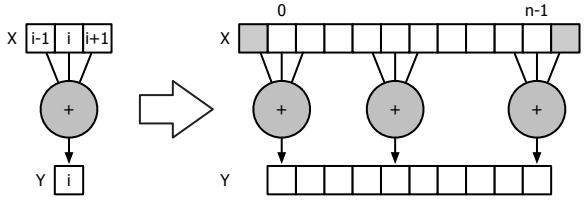

FIGURE 1 – stencil 1D S_3 et son application à un tableau de n cases

La définition du stencil S_3 doit être spécialisée en explicitant les cas particuliers (eq. 2).

$$y(i) = \begin{cases} x(i) + x(i) + x(i+1) & \text{si } i = 0 \\ x(i-1) + x(i) + x(i) & \text{si } i = n-1 \\ x(i-1) + x(i) + x(i+1) & \text{sinon} \end{cases} \tag{2}$$

1. Afin de débogguer les fonctions à coder, on décide d'initialiser le tableau de la façon suivante : $x(i) \leftarrow x_0 + i \times a$. Calculer les formules indiquant ce que contient chaque case $y(i)$.

2. Ecrire un code C simple avec notations tableau et avec les deux cas particuliers à l'intérieur de la boucle.
Fonction `add3_basic0_without(float *X, int n, float *Y)`.

3. Idem, mais avec les deux cas particuliers à l'extérieur de la boucle.
Fonction `add3_basic1_without(float *X, int n, float *Y)`.

4. Ecrire un code C avec *scalarisation* (= mise en variables) correspondant au code précédent.
Fonction `add3_reg_without(float *X, int n, float *Y)`.

5. Ecrire un code C avec *rotation* de registres et *scalarisation*.
Fonction `add3_rot_without(float *X, int n, float *Y)`.

6. Ecrire un code avec déroulage de boucle (afin d'obtenir une *scalarisation* totale). Pour cela, implémenter des déroulages d'ordre 2, 3 et 4 avec prologue mais sans épilogue (dans un premier temps).
Fonctions `add3_lu2_without(float *X, int n, float *Y)`, `add3_lu3_without(float *X, int n, float *Y)` et `add3_lu4_without(float *X, int n, float *Y)`.

7. Une fois trouvé le déroulage minimisant les déplacements de valeurs (entre variables ou depuis la mémoire), ajouter l'épilogue.

Pipeline logiciel (*software pipeline*)

Dans la suite, on suppose que le tableau X a des bords (fig. 2). Les cases `X[-1]` et `X[n]` existent et sont initialisées avec la case d'à coté : `X[-1]=X[0]` et `X[n]=X[n-1]` (pour les questions 1 à 4).

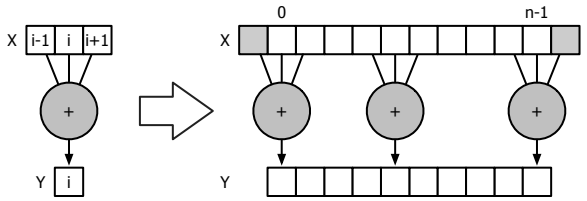

FIGURE 2 – stencil 1D S_3 et son application à un tableau de n cases avec un bord de 1 à gauche et à droite

Dans la partie précédente, on attendait d'avoir chargé tous les points dans des variables pour lancer le calcul en une fois (via la macro `add3`). On lançait donc les calculs *au plus tard*. Dans cette partie, on va lancer les calculs *au plus tôt*, grâce à un pipeline logiciel qui s'étalera sur plusieurs itérations de boucle. Dans un premier temps, on suppose que le tableau a des bords, afin de ne pas complexifier davantage l'algorithme et son code.

Pour chacune des questions, il est demandé de faire deux schémas. Le premier contient pour chaque ligne, x_i le point de X qui est chargé et à coté, les éléments y_p dont qui utilisent x_i dans leur calculs (sur une même ligne, écrire les différents y_j de manière croissante, et que, d'une ligne à l'autre, les différentes utilisations de y_j soient en colonne).

Identifier alors, le corps de boucle (avec une structure régulière de calculs), le prologue (les calculs précédents la première instance du corps de boucle) et l'épilogue (les calculs suivant la dernière instance du corps de boucle). Pour cela, il faut donc écrire suffisamment de lignes pour voir apparaitre deux corps de boucle. Une fois ce travail fait, il faut *scalariser* tous les calculs. Ensuite, il faut refaire un autre schéma (même structure), mais avec des variables $\texttt{x}_\texttt{p}$ et $\texttt{y}_\texttt{q}$ et non avec des éléments de $x(n)$ ou $y(n)$.

1. Comme chaque élément x_i sert trois fois, réaliser un pipeline impliquant le calcul de 3 valeurs de y_j différentes. Il doit donc y avoir 3 lectures et 3 écritures dans le corps de boucle. Fonction `add3_sp3`.
2. Faire un pipeline avec 2 lectures et 2 écritures pour calculer 2 y_j. Fonction `add3_sp2`.
3. Faire un pipeline avec 1 seule lecture et une seule écriture pour calculer 2 y_j. Fonction `add3_sp_rot2`.
4. Dérouler la boucle de la fonction précédente pour obtenir un pipeline avec 2 lectures, 2 écritures et le calcul de 2 y_j. Fonction `add3_sp_lu2`.
5. Refaire de même, avec un tableau sans bord et pour la fonction `add3_sp3_without`.
6. Refaire de même, avec un tableau sans bord et pour la fonction `add3_sp2_without`.
7. Refaire de même, avec un tableau sans bord et pour la fonction `add3_sp_rot2_without`.
8. Refaire de même, avec un tableau sans bord et pour la fonction `add3_sp_lu2_without`.