Compte-rendu – Vectorisation

SFPN1 – AVX2 – gcc

Il y a pour chaque exemple 2 versions OpenMP : une version par défaut (qui vectorise en SSE), et une qui vectorise en AVX2, grâce à l'option -mavx2 de gcc.

Produit terme à terme

La version à vectoriser par le compilateur est vectorisée en SSE avec l'option -O3, son temps rejoint alors celui de la version OpenMP en SSE.
Le temps de la version OpenMP en AVX2 est très proche de celui de la version AVX2 avec les intrinsèques et les 2 sont plus rapides que les versions vectorisées en SSE (grâce à la taille supérieure des vecteurs AVX2 par rapport aux vecteurs SSE).
On a un facteur de 6 entre le temps des versions AVX2 et non vectorisée (facteur théorique de 8).

Produit scalaire

La version à vectoriser par le compilateur n'est pas vectorisée pas même avec l'option -O3. Sa vectorisation n'est pas aussi évidente que celle du produit terme à terme : le code n'est pas directement parallèle à cause de l'accès à chaque itération à la même variable, on a donc besoin en OpenMP d'utiliser la clause reduction.
La version OpenMP AVX2 est vectorisée avec FMA à partir de l'option -O2 et ses temps rejoignent alors ceux de la version avec intrinsèques.
Pour une raison inconnue, le temps de la version OpenMP en SSE est multiplié par 3 entre l'utilisation de -O2 et -O3, et augmente sur la version OpenMP en AVX2 entre la version sans FMA et celle avec.
Il y a un facteur de 4 entre le temps des versions AVX2 et non vectorisée (facteur théorique de 16).
La différence entre les 2 peut être expliqué par la réduction qui ajoute des opérations.

Produit matriciel

La vectorisation par le compilateur ne se fait pas pour la même raison que sur le produit scalaire.
Les versions OpenMP utilisent la clause reduction.
Les versions OpenMP ne sont cependant pas plus rapides que les versions compilateur car il est nécessaire de changer le code pour mieux exploiter la parallélisation.
La première version avec intrinsèques utilise un algorithme différent, afin d'exploiter la vectorisation. Elle est donc plus rapide que les autres versions avec un facteur de 7 (facteur théorique de 16). La différence entre les 2 facteurs est due à l'utilisation de reduction.
La version alternative avec intrinsèques est une version naïve basée sur des produits scalaires successifs entre ligne et colonnes. Elle est plus coûteuse en appels mémoire et provoque des défauts de cache, ce qui explique son temps haut.

Knightslanding – AVX512 – gcc

Produit terme à terme

La version non vectorisée utilise le FMA en scalaire.
Le compilateur vectorise en AVX512 à partir de l'option -O2, la version à vectoriser par le compilateur rejoint alors les temps de la version OpenMP.
Les temps sont inexplicablement différents entre la versions avec intrinsèques et les autres versions vectorisées, mais la différence est faible.
Il y a un facteur de 12.6 entre les versions vectorisées et la version non vectorisée (facteur théorique de 16).

Produit scalaire

La version non vectorisée utilise le FMA en scalaire.
Il y a une grande différence entre les versions vectorisées : la version vectorisée par le compilateur est 2 fois plus rapide que la version OpenMP, elle-même 2 fois plus rapide que la version avec intrinsèques. Je n'ai pas d'explication.
La version vectorisée la plus rapide (par le compilateur) est 15 fois plus rapide que la version non vectorisée (facteur théorique de 16).

Produit Matriciel

La version non vectorisée utilise le FMA en scalaire.
La version OpenMP est la plus lente,
Alors que la version utilisant les intrinsèques est plus efficace en théorie que les autres versions, celle vectorisée par le compilateur est plus rapide sur une matrice de taille N=512. Cela est dû à la petite taille des caches des coeurs sur Knights Landing. Sur une matrice de taille N=256, la puissance de calcul de la version avec intrinsèques est plus grande que celle des autres versions et similaire à la version vectorisée par le compilateur.

Conclusion

Les versions vectorisée sont toujours plus rapides que les versions scalaires correspondantes sauf dans le cas où la vectorisation forcée n'est pas efficace (OpenMP ou la version naïve avec intrinsèques sur le produit matriciel).

Les puissances de calcul obtenues sont bien en dessous des valeurs théoriques :
- Pour sfpn1, la puissance de calcul maximale obtenue est de 10.9 Gflop/s, la puissance théorique du processeur est de 224 Gflop/s, soit une utilisation du processeur inférieure à 5 %.
- Pour knightslanding, la puissance de calcul maximale obtenue est de 6 Gflop/s, la puissance théorique du processeur est de 3456 Gflop/s, soit une utilisation du processeur inférieure à 0.018 %.
Cela est dû notamment au fait que l'on utilise qu'un seul coeur, et que l'architecture de knights landing en particulier est basée sur un grand nombre de coeurs de faible puissance (72).

Quand un code est parallélisable, OpenMP permet de le vectoriser simplement. Les intrinsèques permettent cependant un plus grand contrôle sur la vectorisation.
Le compilateur icc est plus efficace que gcc au niveau de la vectorisation (produit matriciel par exemple).