Software & Programming of HPC Systems

ΕΡΓΑΣΙΑ 2: SIMD and CUDA

Φλουρής Παντελής, up1093507

Κολτσάκης Χρυσάφης, up1084671

1. **SIMD and WENO5**

**A)**

Η ενεργοποίηση των GCC flags **-O -march=native -ftree-vectorize** στον αρχικό κώδικα δεν παρείχε καμια βελτιστοποίηση. Αυτο εύκολα λύθηκε με μετατροπή της weno\_minus\_core σε static inline, επιτρέποντας στον GCC να την κανει vectorize.

Μετά απο αυτή την αλλαγή, ο gcc δημιουργεί δυο εκδόσεις: Μια χρησιμοποιώντας SSE και μία χρησιμοποιώντας AVX, πετυχαίνοντας έτσι μεγαλύτερη συμβατότητα με παλαιότερα συστήματα. Προσθέσαμε και την εντολη **#pragma GCC ivdep** πριν τον βρόγχο επανάληψης, ξεκαθαρίζοντας στον compiler ότι δεν υπάρχουν dependencies.

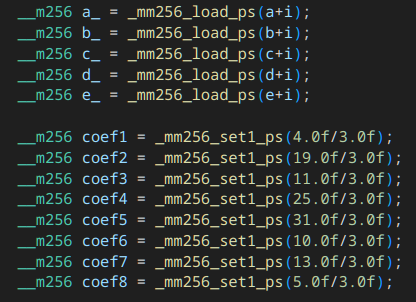
Η χρηση OpenMP vectorization ηταν πιο απλή αλλα λιγότερο αποδοτική. Στον αρχικό πηγαίο κώδικα, καναμε μόνο αυτές τις αλλαγές:

* Προσθήκη <omp.h>
* #pragma omp declare simd στην συναρτηση weno\_minus\_reference
* #pragma omp parallel for simd aligned(a, b, c, d, e, out:64) στο for loop

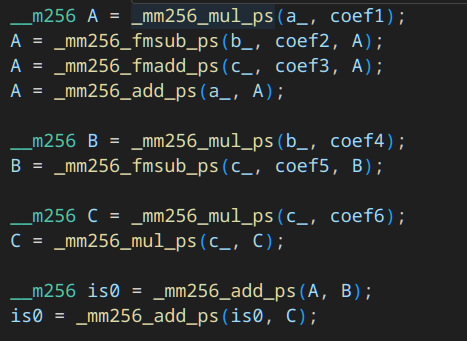
Αξίζει να σημειωθεί οτι η χρηση των OpenMP directives στον κώδικα του πρώτου υποερωτήματος (με τα compiler vectorizations) δεν παρείχε καμια βελτίωση).

Για την παραλληλοποίηση με AVX intrinsics, σβησαμε την συνάρτηση weno\_minus\_core και τοποθετήσαμε όλο τον κώδικα απευθείας μεσα στην weno\_minus\_reference. Αυτο το βήμα δεν είναι απαραίτητο, αλλα το βρήκαμε πιο εύκολο.

Ο βρόγχος επανάληψης γινεται με βήματα των 8, και αρχικοποιούνται σε καταχωρητές 8 τιμές του κάθε float array και μερικοί καταχωρητές με 8 αντίγραφα του κάθε αριθμού που χρειαζόμαστε για τις πράξεις.



Στην συνέχεια υπολογίζονται και αποθηκεύονται σε AVX καταχωρητές οι τιμές is0, is1, is2 με χρήση fmadd και fmsub όπου ειναι εφικτό:



Συνεχίζουμε την αντιστοιχη διαδικασία για το υπόλοιπο του κώδικα. Χρησιμοποιούμε την εντολη \_mm256\_rcp\_ps στα σημεία που χρειαζόμαστε κάποιο αντίστροφο.

Στο τέλος αποθηκεύουμε τα αποτελέσματα στην μνήμη.

**Β)**

Αποτελέσματα:

Ο πιο γρήγορος χρόνος επιτεύχθηκε με την χρήση GCC flags. Υποψιαζόμαστε οτι ο λόγος που είναι σημαντικά γρηγορότερος ο κώδικας ειναι πως η χρήση O3 κάνει περισσότερα απο αυτόματο vectorization, και αυτά ευθύνονται για την βελτίωση στην ταχύτητα.

1. **Parallel Parametric Search in Machine Learning**