Παραλληλοποίηση Με OpenACC

Μεταφορά δεδομένων εισόδου στην GPU

Οι εικόνες βρίσκονται απιθηκευμένες στον πίνακα `input[NUM_IMAGES][IMAGE_PIXELS]`. Η δέσμευση μνήμης για τον πίνακα γίνεται:

```
float ** input =(float **)malloc(NUM_IMAGES*sizeof(float *));
   for (int i = 0 ; i < NUM_IMAGES ; i++){
       input[i]=(float*)malloc(sizeof(float)*IMAGE_PIXELS);
   }

   load_data(input,labels,NUM_IMAGES);

#pragma acc enter data copyin (input[:NUM_IMAGES])
   for (int i = 0 ; i < NUM_IMAGES ; i++){
#pragma acc enter data copyin (input[i][:IMAGE_PIXELS])
   }
   /* ... */
  for(int i=0;i<NUM_IMAGES;i++){
       free(input[i]);
#pragma acc exit data delete(input[i])
   }
   free(input);
#pragma acc exit data delete(input)
```

> [!NOTE] Δεν επιλέχθηκε συνεχής δέσμευση μνήμης για το `input`, καθώς είναι δύσκολο για την **openacc** να κάνει την αντιστοίχιση των μεταβλητών.

Το `#pragma acc enter data create` δεσμεύει χώρο για την ίδια μεταβλητή στην GPU. Για κάθε malloc που γίνεται στον host, γίνεται το αντίστοιχο data create στο device. Αντίστοιχα στο τέλος του προγράμματος, σε κάθε free αντιστοιχεί ένα `#pragma acc exit data delete`.

Για την αντιγραφή των δεδομένων από το device στον host και αντίστροφα χρησιμοποιούμε `update device` και `update self` αντίστοιχα.

Το `#pragma acc enter data copyin` δεσμεύει χώρο και αντιγράφει τις τιμές που έχουν οι μεταβλητές.

Ελέγχω ότι τα δεδομένα φορτώνονται σωστά στην GPU, προσθέτοντας 1 σε κάθε pixel της εικόνας

```
#pragma acc parallel loop present(input)
    for (int i=0;i<NUM_IMAGES;i++){
        for (int j=0;j<IMAGE_PIXELS;j++){
            input[i][j]+=1;
        }
    }
for (int i=0;i<NUM_IMAGES;i++){
```

```
        #pragma acc update self(input[i][:IMAGE_PIXELS])
    }
```

> [!CAUTION] To `#pragma acc parallel loop copyout(input[:NUM_IMAGES][:IMAGE_PIXELS])` δεν επιστρέφει τα δεδομένα στον host. Χρειάζεται `update self`.