

Relazione Homework 2: Matrice

A cura di Sadman Sakib Rahman, Tiziano Natali, Luca Mancini

Matricole: 1632174, 1554933, 1649441

L'IDEA

L'approccio algoritmico utilizzato per effettuare l'estensione da matrici quadrate a matrici rettangolari riadatta l'algoritmo visto a lezione per effettuare la stessa operazione su matrici quadrate. L'idea consiste nell'eseguire in maniera parallela le operazioni di ogni singola cella, e, successivamente, nell'algoritmo che usufruisce della memoria locale, effettuare le stesse operazioni in una sottomatrice interna di dimensioni fisse.

L'IMPLEMENTAZIONE

Per modificare il programma originale della moltiplicazione fra matrici abbiamo innanzitutto aggiunto una nuova variabile globale nel codice host tramite `#define` per rappresentare la dimensione del lato corto del rettangolo.

Abbiamo quindi cambiato gli indici utilizzati per il calcolo, ricordandoci che una matrice di dimensioni $n \times k$ moltiplicata per una matrice di dimensioni $k \times n$, è uguale a una matrice $n \times n$.

Abbiamo inoltre cambiato i valori per l'allocazione in memoria tramite `malloc` di A e B, e i parametri nel metodo `clCreateBuffer` dei MEM OBJECT in maniera da rispecchiare la grandezza delle matrici rettangolari.

Nel kernel abbiamo aggiunto una nuova variabile in input, le dimensioni del lato corto. Infine, abbiamo cambiato il codice sequenziale affinché potesse effettuare moltiplicazioni fra matrici rettangolari.

Nel programma con utilizzo della memoria locale dei workgroups, per poter eseguire il programma con matrici rettangolari a prescindere dalla variabile `LOCAL_SIZE`, abbiamo approssimato le dimensioni dei lati al valore del primo multiplo della `LOCAL_SIZE` maggiore delle dimensioni originali.

PIATTAFORME

MacBook Pro di Tiziano Natali



macOS Sierra
Version 10.12.2

MacBook Pro (Retina, 13-inch, Late 2013)
Processor 2.6 GHz Intel Core i5
Memory 8 GB 1600 MHz DDR3
Startup Disk Macintosh HD
Graphics Intel Iris 1536 MB
Serial Number C02LT173FH01

MacBook Pro di S. Sakib Rahman



macOS Sierra
Versione 10.12.2

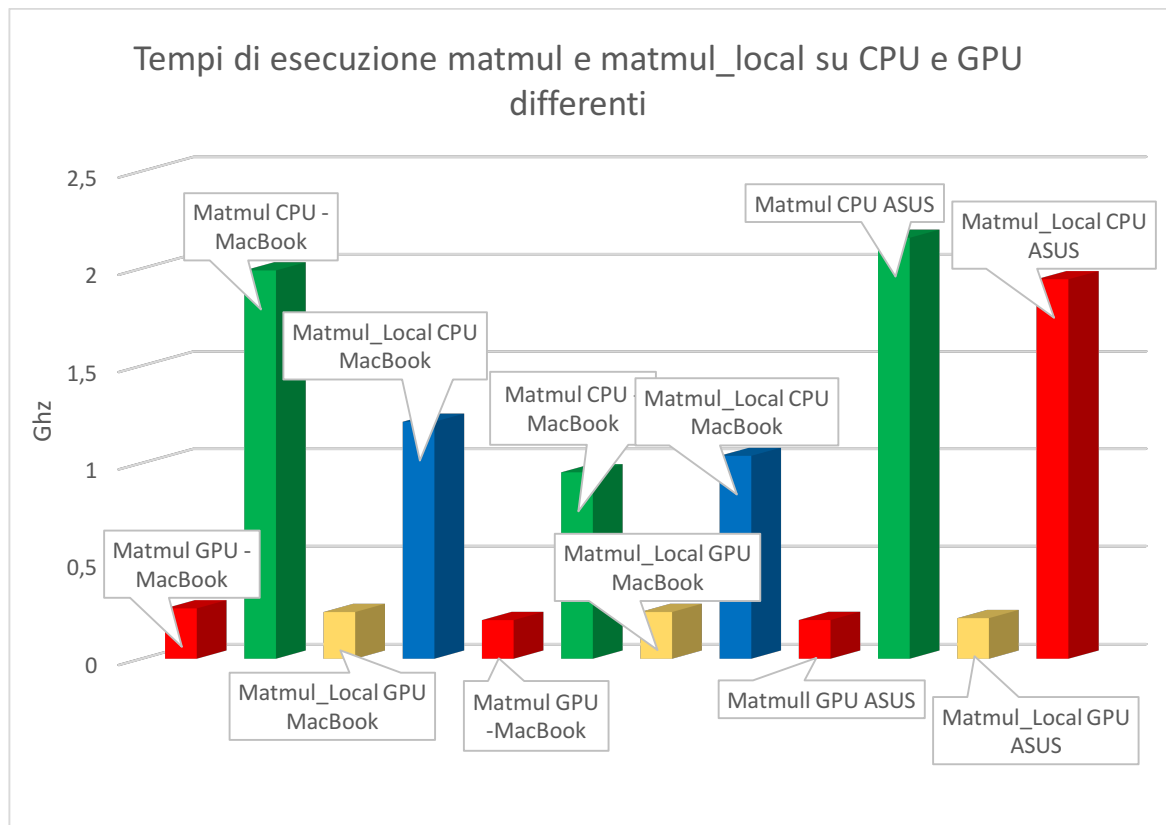
MacBook Pro (Retina, 13-inch, Late 2013)
Processore 2,4 GHz Intel Core i5
Memoria 4 GB 1600 MHz DDR3
Disco di avvio Ridam
Scheda grafica Intel Iris 1536 MB
Numero di serie C02LJ706FGYY

ASUS TP300LA di Luca Mancini

Intel i5-4210U CPU @ 1.70GHz 2.40
RAM 8.00 GB
Sistema operativo 64-bit
Model: ASUS TP300LA
WINDOWS 10
GPU- INTEL HD Graphics 4400

DATI SUGLI ESPERIMENTI

Paragone tra tempo di esecuzione parallela e sequenziale su input diversi



CONSIDERAZIONI FINALI

Abbiamo constatato che all'aumentare delle dimensioni dei lati delle matrici il codice che utilizza memoria locale risulta essere il più efficiente. Inoltre all'aumentare del divario fra i lati, l'esecuzione su GPU risulta essere molto più veloce che su CPU.

ISTRUZIONI PER ESECUZIONE DA TERMINALE

`cd DIRECTORY/LOCATION_OF_C_FILE`

`make NAME_OF_C_FILE_WITHOUT_C_EXTENSION`

`./NAME_OF_C_FILE_WITHOUT_C_EXTENSION`

Esempio di esecuzione:

```
Last login: Tue Jan 10 17:59:06 on ttys000
MacBook-Pro-di-Ridam:~ Ridam$ cd /Users/Ridam/Downloads/Università/3\ Anno/Progettazione\ di\ Sistemi\ Multicore/openCL_examples/matmul2
MacBook-Pro-di-Ridam:matmul2 Ridam$ make matmul
gcc -O3 -framework OpenCL -I../clut/ matmul.c ../clut/clut.o -o matmul
MacBook-Pro-di-Ridam:matmul2 Ridam$ ./matmul
Device info:
  Name:      Iris
  Vendor:    Intel
  OpenCL version: OpenCL C 1.2
  Available: Yes
  Compute units: 40
  Clock frequency: 1100 Mhz
  Global memory: 1536 MB
  Max allocatable memory: 384 MB
  Local memory: 65536 KB
  Max work group size: 512
data size: 1000
global size: 1000
Tempo esecuzione su GPU: 0.197685 sec
Tempo esecuzione su CPU: 1.194693 sec
correctness test: PASSED
MacBook-Pro-di-Ridam:matmul2 Ridam$
```