Sistemi di Calcolo Parallelo e Applicazioni: Progetto di fine corso

Problema

Il progetto verte sulla realizzazione di un nucleo di calcolo per il prodotto tra una matrice sparsa ed un vettore, che sia in grado di calcolare

$$y \leftarrow Ax$$

dove A è una matrice sparsa memorizzata nei formati

1. CSR;
2. HLL;

Il nucleo dovrà essere parallelizzato per sfruttare le risorse di calcolo disponibili, con parallelizzazione OpenMP e CUDA, e potrà essere sviluppato in C; inoltre sarà collaudato funzionalmente confrontando con una implementazione seriale minimale (per la quale si suggerisce di usare un formato CSR). Sarà necessario sviluppare delle funzioni ausiliarie per il preprocessamento dei dati della matrice e per la memorizzazione nel formato desiderato. Sarà necessario misurare le prestazioni del codice, sul server disponibile in dipartimento

Matrici di test

Per il collaudo si useranno delle matrici disponibili dalla "Suite Sparse Matrix Collection" all'indirizzo `https://sparse.tamu.edu/`. Si raccomanda di scaricare le matrici nel formato MatrixMarket; all'indirizzo `http://math.nist.gov/MatrixMarket/` sono disponbili delle funzioni per I/O.

Le matrici sono caratterizzate da numero di righe M, numero di colonne N e numero di nonzeri NZ. Normalmente, per le matrici simmetriche solo un triangolo (superiore o inferiore) è memorizzato su file; ai fini del progetto, si deve invece assumere che in memoria siano memorizzate completamente, perciò all'atto della lettura da file si deve predisporre la ricostruzione esplicita del triangolo mancante, con ricalcolo del valore NZ. Alcune delle matrici sono memorizzate come "pattern"; in questo caso tutti i coefficienti nonzero valgono 1.0 e vengono quindi sottintesi nella memorizzazione su file per risparmiare spazio su disco.

Il collaudo comprenderà almeno le seguenti matrici:

cage4	Cube_Coup_dt0	FEM_3D_thermal1
mhda416	ML_Laplace	thermal1
mcfe	bcsstk17	thermal2
olm1000	mac_econ_fwd500	thermomech_TK
adder_dcop_32	mhd4800a	nlpkkt80
west2021	cop20k_A	webbase-1M
cavity10	raefsky2	dc1
rdist2	af23560	amazon0302
cant	lung2	af_1_k101
olafu	PR02R	roadNet-PA

Rimane sempre possibile effettuare ulteriori prove con altre matrici.

Misurazione delle prestazioni

Tutte le misure saranno ottenure con la invocazione ripetuta del nucleo di calcolo per un certo numero di volte per ciascuna matrice, producendo così una misura del tempo medio per invocazione.

In tutti i casi la misura in MFLOPS o GFLOPS sarà calcolata come

$$FLOPS = \frac{2 \cdot NZ}{T}$$

dove NZ è il *numero di nonzeri* nella matrice, corretto se necessario nel caso di memorizzazione simmetrica su file, ma *senza* considerare eventuali zeri aggiuntivi che si rendono necessari in alcuni formati, e T è il tempo (medio) di esecuzione.

Le misure includeranno solo il tempo necessario al calcolo del prodotto matrice vettore; il preprocessamento dei dati, il tempo di I/O ed il tempo necessario al trasferimento dati da e verso la scheda CUDA *non* devono essere inclusi (ma possono essere misurati e discussi a parte).

Per la versione OpenMP, il codice sarà collaudato con un numero di thread variabile da 1 fino al massimo numero di core disponibili sulla piattaforma di calcolo.

Formati di memorizzazione

I formati di memorizzazione vengono qui descritti con un codice Matlab.

Nella traduzione in C/CUDA, si ponga particolare attenzione all'accesso ai dati ed agli aggiustamenti alle strutture dati che possono essere necessari per tenere conto del fatto che Matlab memorizza le matrici bidimensionali per colonne, mentre C e C++ per righe.

CSR

Compressed Storage by Rows. Una matrice $M \times N$ con NZ non-zeri viene descritta con i seguenti dati:

M Numero di righe;

N Numero di colonne;

IRP(1:M+1) Vettore dei puntatori all'inizio di ciascuna riga;

JA(1:NZ) Vettore degli indici di colonna;

AS(1:NZ) Vettore dei coefficienti;

di modo che il codice del prodotto in Matlab sarebbe:

```
for i=1:m
  t=0;
  for j=irp(i):irp(i+1)-1
     t = t + as(j)*x(ja(j));
  end
  y(i) = t;
end
```

Ad esempio, la matrice

$$\begin{pmatrix} 11 & 12 & 0 & 0 \\ 0 & 22 & 23 & 0 \\ 0 & 0 & 33 & 0 \\ 0 & 0 & 43 & 44 \end{pmatrix}$$

verrebbe memorizzata in CSR (assumendo indici in base 1 utilizzati in Matlab) come:

$$\begin{aligned} M &= 4 \\ N &= 4 \\ IRP &= [1, 3, 5, 6, 8] \\ JA &= [1, 2, 2, 3, 3, 3, 4] \\ AS &= [11, 12, 22, 23, 33, 43, 44] \end{aligned}$$

HLL

Il formato HLL può essere definito informalmente in questo modo:

- Si stabilisce un parametro HackSize (p.es. 32);
- Si partiziona la matrice di input in blocchi di HackSize righe;
- Ciascun blocco viene memorizzato in formato ELLPACK (descritto nel seguito);
- I blocchi vengono memorizzati in una struttura dati da determinare.

ELLPACK

ELLPACK storage format.

Una matrice $M \times N$ con NZ non-zeri e tale che $MAXNZ$ sia il massimo numero di nonzeri per riga, su tutte le righe, viene descritta come segue:

M Righe;

N Colonne;

MAXNZ Max nonzeri per riga;

JA(1:M,1:MAXNZ) 2D array di indici di colonna;

AS(1:M,1:MAXNZ) 2D array di coefficienti;

così che il prodotto in Matlab verrebbe calcolato come segue:

```
for i=1:m
  t=0;
  for j=1:maxnzr
    t = t +  as(i,j)*x(ja(i,j));
  end
  y(i) = t;
end
```

Le righe che abbiano un numero di nonzeri inferiore a $MAXNZ$ devono essere riempite con coefficienti appropriati, ossia zeri in AS. Ad esempio la matrice

$$\begin{pmatrix} 11 & 12 & 0 & 0 \\ 0 & 22 & 23 & 0 \\ 0 & 0 & 33 & 0 \\ 0 & 0 & 43 & 44 \end{pmatrix}$$

può essere memorizzata in ELLPACK (assumendo indici a base 1 in Matlab) come:

$$\begin{aligned} M &= 4 \\ N &= 4 \\ MAXNZ &= 2 \\ JA &= \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 3 \\ 3 & 4 \end{pmatrix} \\ AS &= \begin{pmatrix} 11 & 12 \\ 22 & 23 \\ 33 & 0 \\ 43 & 44 \end{pmatrix} \end{aligned}$$

In questo caso si rende necessario un solo zero aggiuntivo (ed il corrispondente indice in `JA` viene fissato all'ultimo indice valido incontrato lungo la riga).

La descrizione del formato ELLPACK in Matlab richiede attenzione per il fatto che la memorizzazione per colonne implica che i coefficienti di ciascuna colonna siano adiacenti in memoria. Per riprodurre l'adiacenza in un linguaggio con memorizzazione per righe, i dati della matrice dovrebbero "apparire" trasposti (anche se questo corrisponderebbe allo stesso layout in memoria).

Ad esempio in CUDA, se si assegna un thread per riga, gli accessi coalizzati ai dati richiedono un layout appropriato in memoria (i.e. trasposto).

Note

Il materiale da presentare al termine del progetto consiste in:

1. Il codice compilabile;
2. Una relazione sul lavoro effettuato, che includa una analisi delle prestazioni ottenute.

È possibile sviluppare il progetto in gruppi da non più di due persone.

Altre osservazioni:

- Il codice può essere sviluppato su qualunque piattaforma, ma i dati di performance devono essere raccolti sul server in dipartimento;
- Il codice dovrà quindi essere compilabile sul server di dipartimento;
- La valutazione dell'esame si baserà su:
  - Livello di prestazioni ottenute;
  - Qualità complessiva della relazione;
  - Qualità della analisi delle prestazioni.
- Coloro che decidessero di presentare un elaborato di gruppo dovranno indicare brevemente nella relazione la suddivisione del lavoro tra i componenti del gruppo stesso