University of Calabria, DIMES
High Level Synthesis of Digital Systems
2023-2024
Prof.ssa PERRI
Prof. FRUSTACI
Sparse Matrix Vector Multiplication Analysis

Giorgio Ubbriaco
247284
bbrgrg00h11d086x@studenti.unical.it

June 2024

Index

Listings

List of Figures

List of Tables

1 Introduction

1.1 Sparse Matrix

Nell'analisi numerica, una **matrice sparsa** è una matrice in cui la maggior parte degli elementi è pari a zero. Non esiste una definizione rigorosa della proporzione di elementi a valore nullo affinché una matrice possa essere considerata sparsa. Al contrario, se la maggior parte degli elementi è non nulla, allora la matrice è considerata densa.
Una matrice è tipicamente memorizzata come un array bidimensionale. Ogni voce della matrice rappresenta un elemento $a_{i,j}$ della matrice e vi si accede tramite i due indici i e j. Per una matrice m $\times$ n, la quantità di memoria necessaria per memorizzare la matrice in questo formato è proporzionale a m $\times$ n (senza considerare che è necessario memorizzare anche le dimensioni relative alla matrice).
Nel caso di una matrice sparsa, è possibile ridurre notevolmente i requisiti di memoria memorizzando solo le voci non nulle. A seconda del numero e della distribuzione delle voci non nulle, è possibile utilizzare diverse strutture di dati che consentono di ottenere enormi risparmi di memoria rispetto all'approccio di base. Il compromesso è che l'accesso ai singoli elementi diventa più complesso e sono necessarie strutture aggiuntive per poter recuperare la matrice originale senza ambiguità.
I formati possono essere divisi in due gruppi:

- Quelli che supportano una modifica efficiente, come DOK (Dictionary of Keys), LIL (List of Lists) o COO (Coordinate List), utilizzati solitamente per la costruzione della matrice.
- Quelli che supportano l'accesso e le operazioni matriciali efficienti, come CRS (Compressed Row Storage) o CCS (Compressed Column Storage).

1.2 Compressed Row Storage (CRS)

Il formato **Compressed Row Storage (CRS)** permette la rappresentazione di una matrice tramite tre array unidimensionali consentendo un accesso veloce alle righe e una moltiplicazione matrice-vettore efficiente. In particolare, i tre array utilizzati sono i seguenti:

- values
  È un array contenente tutti gli elementi della matrice non nulli.
- iFirstEl
  È un array contenente gli indici, relativi all'array values, corrispondenti ai primi elementi non nulli di ogni riga. Nella letteratura questo array è conosciuto anche con la denominazione di *rowPtr*.
- iNonZeroEl
  È un array contenente gli indici di colonna degli elementi non nulli. Nella letteratura questo array è conosciuto anche con la denominazione di *columnIndex*.

2 Tasks to be performed

3 Definitions

4 C Simulations

5 Solutions

6 Conclusions