AVX-512 implementation of GEMM for structured sparse matrices

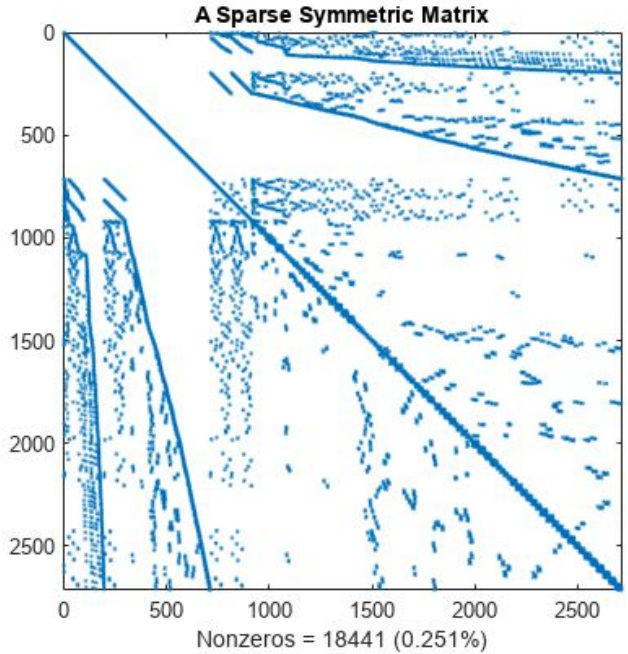
Školitel: Mgr. Vladimír Boža, PhD.
Študent: Jozef Číž

Násobenie matíc

- násobenie matice maticou (obrázky)
- násobenie matice vektorom (text)

Riedka matica

- nie každé číslo je rovnako dôležité
- veľa prvkov vynechaných + malé úpravy = rovnaký výsledok
- matica, ktorá obsahuje veľa nulových prvkov
- matematické operácie sa dajú vykonať efektívnejšie



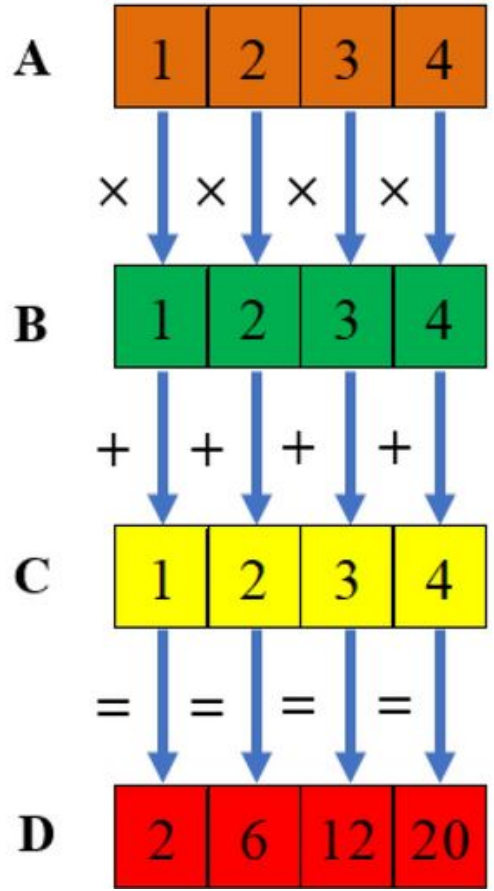
Formáty riedkych matíc

- neštruktúrovaný formát
  - potrebná veľká riedkosť
  - úspora miesta
  - ťažké spraviť to rýchlejšie
  - problémy so stratou schopnosti
- štruktúrovaný formát
  - aj s rozumnou riedkosťou vieme dosiahnuť zrýchlenie
  - pri rozumnej riedkosti sú menšie problémy so zachovaním schopnosti

$$D = \_mm256\_fmadd\_pd(A, B, C)$$

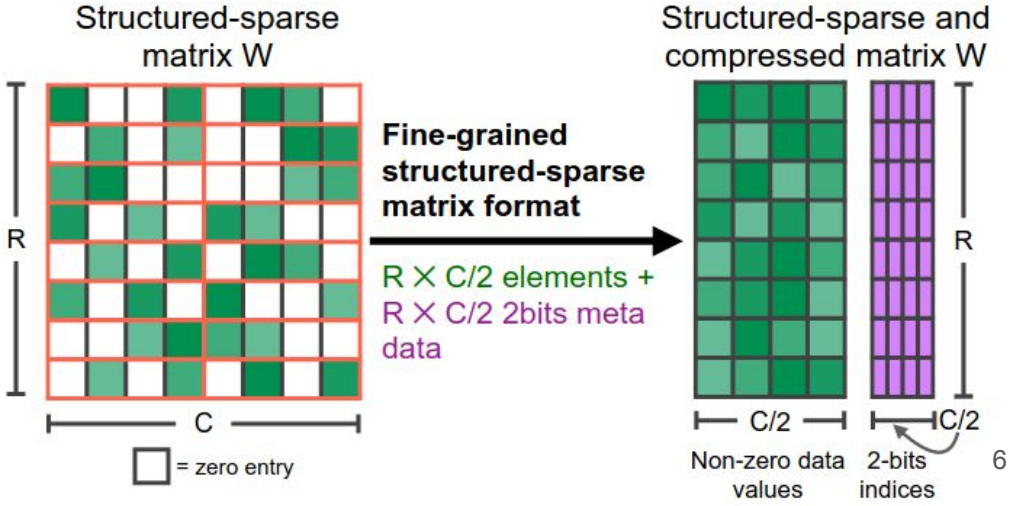
SIMD

- Single Instruction Multiple Data
- vektorové inštrukcie umožňujú vykonať viac operácií naraz
- AVX512
- vynásobenie a sčítanie 16 floatov naraz



Ciele mojej práce

- implementácia násobenia štruktúrovanej riedkej matice hustým vektorom/maticou
- zvýšenie výkonu a úspora pamäte so zachovaním schopnosti
- 1xN s 50% riedkosťou



Existujúce riešenia

- populárne implementácie násobenia hustá*hustá v BLAS
  - OpenBLAS, ATLAS, MKL
- populárne implementácie násobenia riedka*hustá
  - MKL, AOCL
- na základe iných prác vieme povedať, že:
  - je možné prekonať MKL implementáciu riedkeho násobenia na procesore
  - štruktúra 1xN s 50% riedkosťou má dobré vlastnosti
  - bloky 1xN vs 2xN vs 4xN nemajú veľký vplyv na schopnosť

Ako rýchlo násobiť matice?

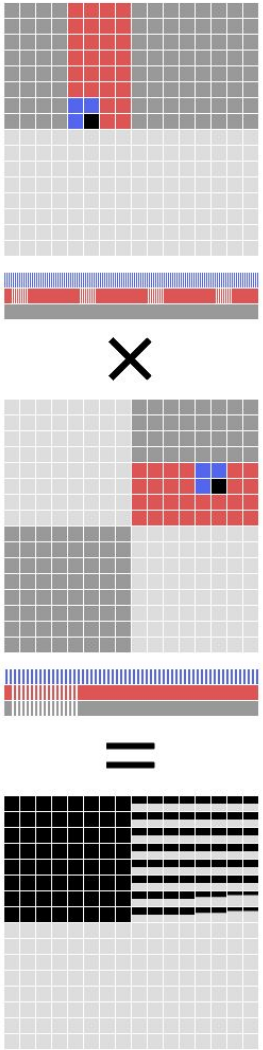
- naivné - n^3 operácií násobenia
- na základe taktu procesoru a špecifikácie architektúry procesora vieme vypočítať teoretický maximálny výkon
  - frekvencia 5.5GHz * priepustnosť 1 = 5.5 GFLOP
- naivná implementácia má iba 0.9 GFLOP
- OpenBLAS má jednovláknový výkon 82 GFLOP

Výkon je obmedzený rýchlosťou počítania

- kľúčové je využívať SIMD inštrukcie
  - maximálny výkon x16 = 88 GFLOP
- naivná implementácia, ktorá používa SIMD inštrukcie má stále iba 5 GFLOP

Výkon je obmedzený rýchlosťou pamäte

- druhým kľúčovým faktorom je správne narábanie s pamäťou
- transponovanie
- kernelovanie => recyklovať registre
- rozdelenie matíc do blokov => recyklovať cache pamäť

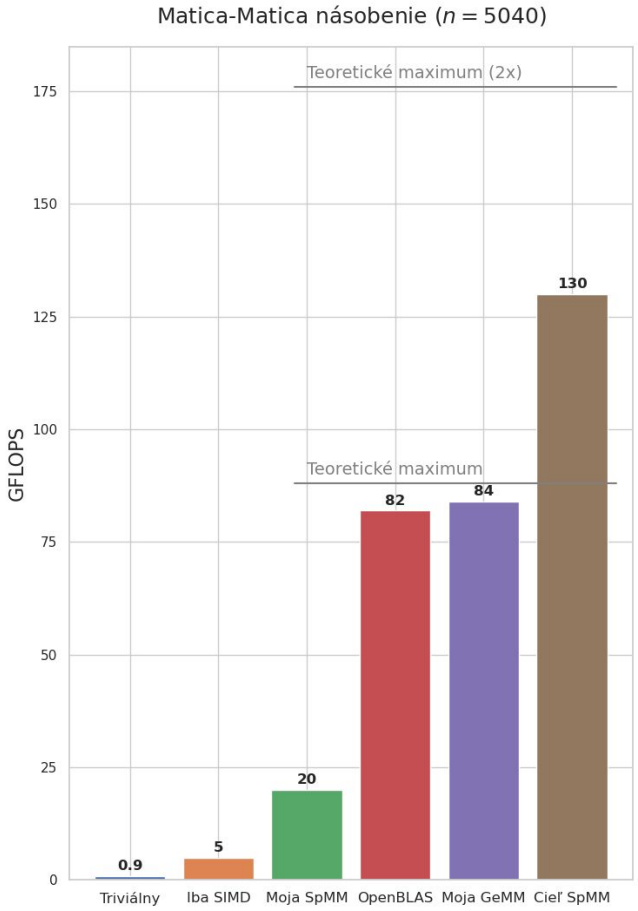


Výsledky - Hustá x Hustá

- OpenBLAS - kernely ručne napísané v assembleri
  - je ťažké takéto kernely napísať a adaptovať na naše potreby
- implementoval a vyladil som jednoduchšiu verziu
  - na mojej architektúre dokonca jemne rýchlejšia než OpenBLAS
  - použijem ju ako podklad pre násobenie riedkych matíc

Výsledky - Riedka x Hustá

- keďže matica má 50% riedkosť, teoretický maximálny výkon je 2x vyšší
- aktuálna MV - jemne rýchlejšia než hustá OpenBLAS
- aktuálna MM - 20 GFLOP
- spôsob akým boli matice delené do blokov v hustom násobení sa ukázal ako nevhodný a treba ho preusporiadať
- cieľom je dosiahnuť MM výkon aspoň 120 GFLOP



Ďalší výskum

- overiť zachovanie schopnosti na maticiach s reálnymi dátami
- zlepšiť testovanie
  - viaceré veľkosti matíc
  - cold/hot cache
- porovnať s MKL/AOCL/CSR5/CVR/SPC5/...

Zrýchlenia

- optimalizovať malé matice
- optimalizovať veľké matice
  - 2 iterácie Strassenovho algoritmu teoreticky vedia ušetriť 25% operácií násobenia
- násobenie matíc je pomerne ľahko dobre paralelizovateľné
  - OpenBlas ~1.1 TFLOP (13.6x na 16T n=5k)