

Počítač: Procesor AMD Ryzen 7 5800HS @ 3.20GHz, RAM 16GB.
Operační systém: Windows 10, Version 10.0.19044 Build 19044
Verze pythonu: 3.8.10
Seed: 94

1 Abstrakt

Představení a porovnání dvou implementací transponování matice. Měří se počet netrefení se do cache a to pro obě implementace a to pro tyto velikosti cache:

- 4 kb (bloky po 16 položkách)
- 32 kb (bloky po 64 položkách)
- 256 kb (bloky po 256 položkách)
- 256 kb (bloky po 4096 položkách)

Implementace máme:

- naivní
- cache friendly

2 Testované implementace

2.1 Standardní naivní implementace

První implementace ignoruje cache a šahá do paměti bez rozmýšlení.

2.2 Chytrá cache friendly implementace

Druhá implementace přistupuje do paměti tak, aby co nejefektivněji využila cache.

3 Měření času

Všechna měření probíhala na stejném zařízení.

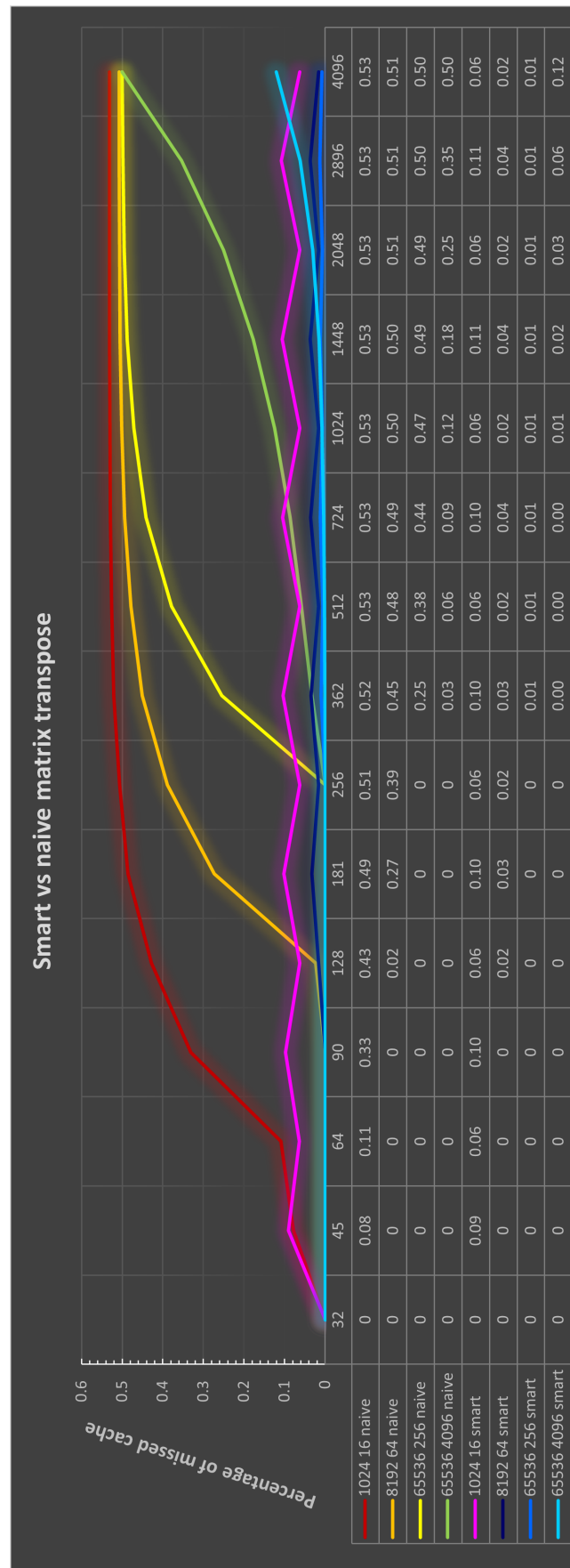
Naivní implementace běžela 3 minuty a 1 sekund (na 16ti vláknech).

Chytrá implementace běžela 2 minuty a 58 sekund (na 16ti vláknech).

Takže je jasně vidět, že python jakožto programovací jazyk je pomalý sám o sobě.

3.1 Shrnutí

Podle analýzy přístupů do paměti jsme zjistili, že chytrá implementace o dost lépe využívá cache, bohužel, to na času běhu programu, nebylo vůbec poznat.



Obrázek 1: Porovnání dvou implementací s různou velikostí cache.