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
- chache friendly

2 Testované implementace

2.1 Standardní naivní implementace

První implementace ignoruje cache a šahá do paměti bez rozmýšlení.

Z grafu níže vidíme, že miss cache byla, u případů kdy cache byla menší než data, cca 50%.

2.2 Chytrá cache friendly implementace

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

Z grafu níže vidíme, že pro každou velikost cache a dat byla cache úspěšně využita průměrně 90% času.

Zajímavá je též oscilace testu "smart 1024 16"kde matice s velikosti dělitelnou 16ti pracují efektivněji, než ty s "nehezkou"velikostí.

3 Měření času na reálném hardware

Všechna měření probíhala na stejném zařízení.

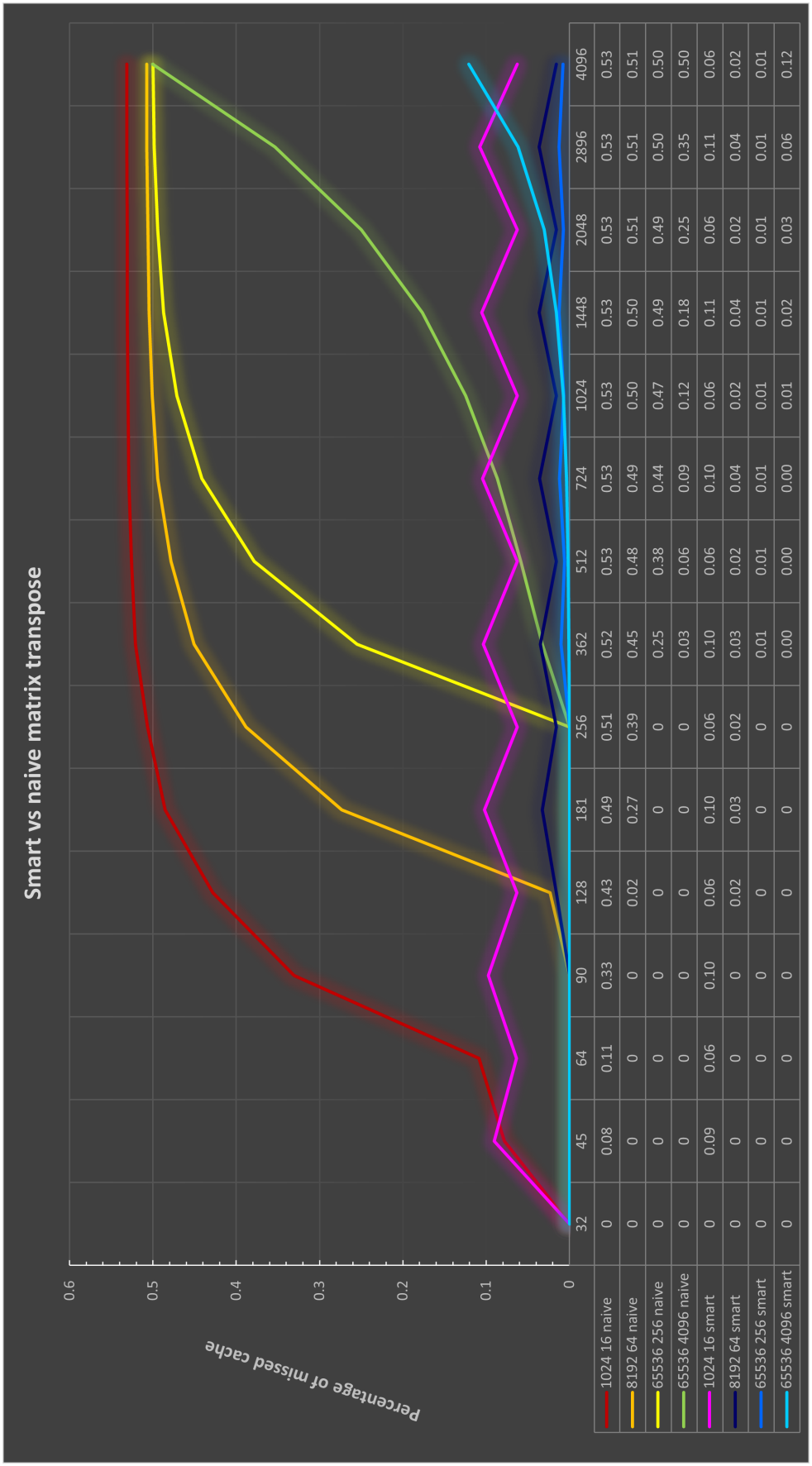
Naivní implementace běžela 3 minuty a 1 sekund (na 4 vláknech).

Chytrá implementace běžela 2 minuty a 58 sekund (na 4 vláknech).

Takže je jasně vidět, že python jakožto programovací jazyk je pomalý sám o sobě. (přesně jak bylo předpokládáno)

3.1 Shrnutí

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



	32	45	64	90	128	181	256	362	512	724	1024	1448	2048	2896	4096
1024 16 naive	0	0.08	0.11	0.33	0.43	0.49	0.51	0.52	0.53	0.53	0.53	0.53	0.53	0.53	0.53
8192 64 naive	0	0	0	0	0.02	0.27	0.39	0.45	0.48	0.49	0.50	0.50	0.51	0.51	0.51
65536 256 naive	0	0	0	0	0	0	0	0.25	0.38	0.44	0.47	0.49	0.49	0.50	0.50
65536 4096 naive	0	0	0	0	0	0	0	0.03	0.06	0.09	0.12	0.18	0.25	0.35	0.50
1024 16 smart	0	0.09	0.06	0.10	0.06	0.10	0.06	0.10	0.06	0.10	0.06	0.11	0.06	0.11	0.06
8192 64 smart	0	0	0	0	0.02	0.03	0.02	0.03	0.02	0.04	0.02	0.04	0.02	0.04	0.02
65536 256 smart	0	0	0	0	0	0	0	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
65536 4096 smart	0	0	0	0	0	0	0	0.00	0.00	0.00	0.01	0.02	0.03	0.06	0.12

Obrázek 1: Porovnání dvou implementací s různou velikostí cache.