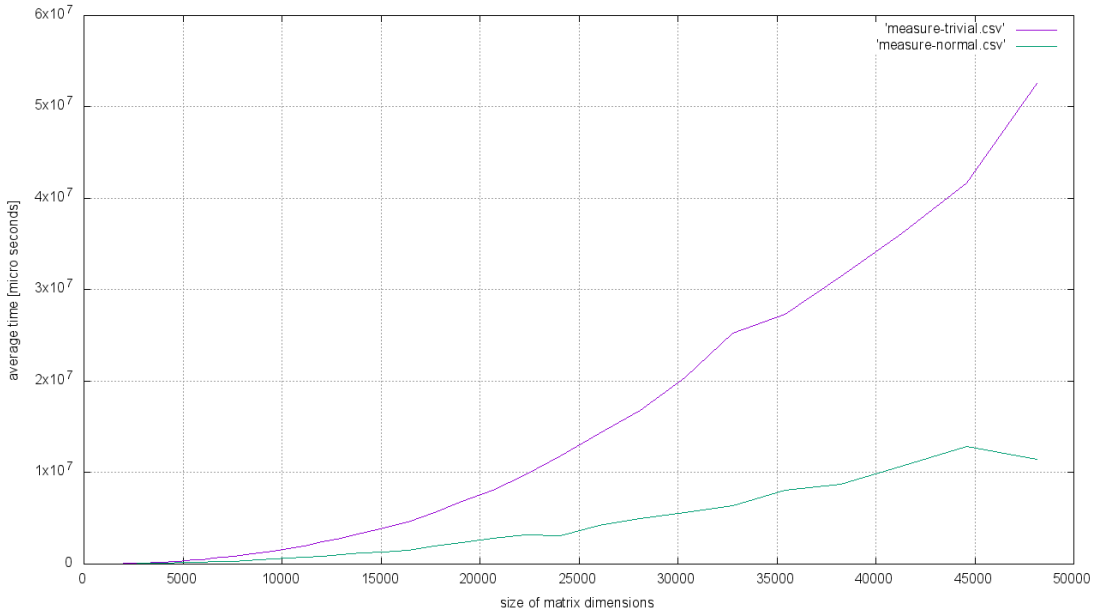
0.1 Triviální versus Cache-oblivious implementace



Pro měření bylo použito až 10 GB paměti RAM. Použitý počítač je osazen procesorem intel i7-920 @2.66 GHz s 64-bitovými instrukecemi AMD64 a následujícími parametry cache:

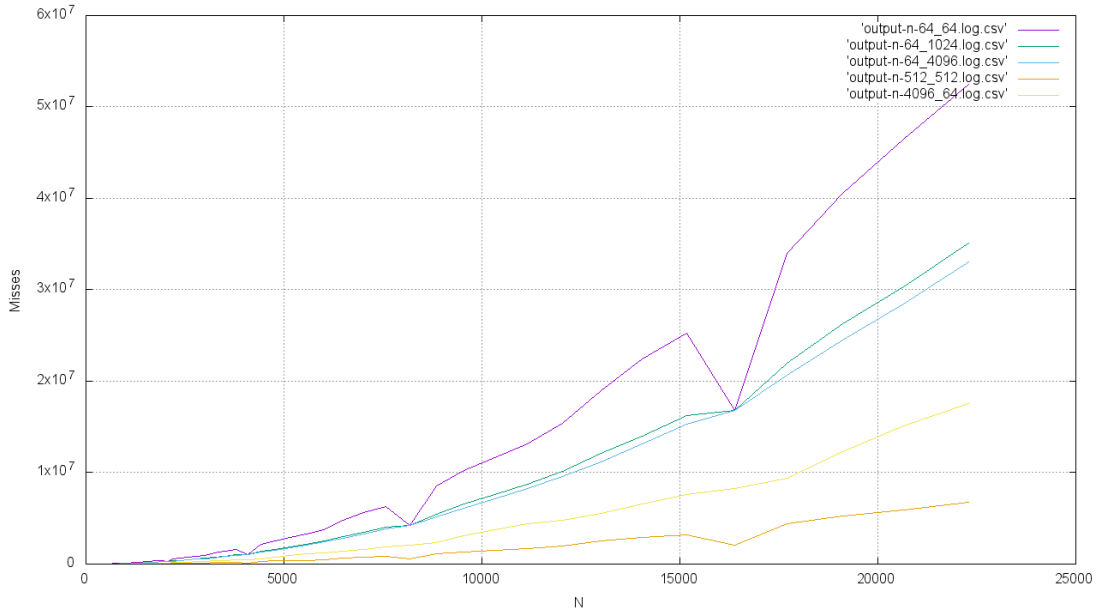
- L1 cache – 32 kB,
- L2 cache – 256 kB,
- L3 cache – 8 MB.

Z grafu vidíme, že triviální implementace je o dost pomalejší. Zároveň vidíme, že pro větší matice je vždy toto zpomalení větší. V následujících bodech dochází u triviální implementace, ke zrychlení růstu, zatímco u cache-oblivious algoritmu tato změna není pozorována.

- 850 MB – N=15000
- 4,56 GB – N=35000
- 7,54 GB – N=45000

Zvláštní je propad cache-oblivious algoritmu v posledním měřeném bodě, nejsem schopný odhadovat, čím je způsoben.

0.2 Simulace cache-oblivious algoritmu na ideální cache



Simulace jsme opět prováděly na stejných datech, jako při měření na počítači, nicméně pouze do maximalní velikosti matice 2 GB. Předchozí graf znázorňuje závislost missů cache na počtu přístupů cache-oblivious algoritmu.

Vidíme, že vítězem byla cache s největší kapacitou, konkrétně cache s velikostí bloku 512 B a počtem bloků 512. Zvýšení velikosti bloků na úkor jejich počtu (4096,64) vedlo k o něco horším výsledkům. Naopak snížení velikosti bloků ve prospěch počtu bloků (64, 4096) vedlo opět ke zpomalení. Dále už je pořadí podle celkove velikosti cache. Vidíme tak, že je třeba volit vyvážený poměr velikosti bloků a jejich počtu, aby cache fungovala co nejlépe.

Opět mi nejsou jasné pravidelné propady doby, které se vyskytují u všech typů cache. Nicméně nejvíce jsou pozorovány u cache s nejmenší kapacitou.