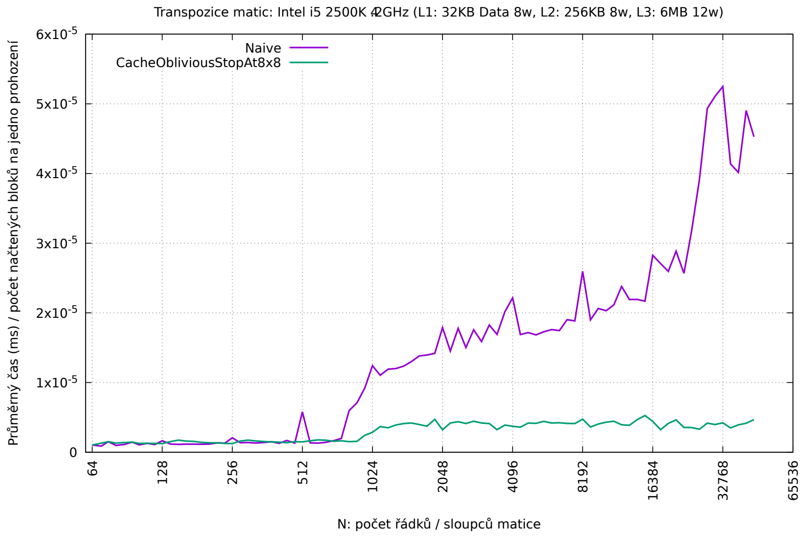
0.1 Reálné CPU



Na prvním grafu vidíme porovnání průměrného času na prohození dvou prvků pro naivní algoritmus a rekurzivní cache oblivious algoritmus, který se zastaví na podmatici velikosti 8x8 a dál pokračuje naivně.

0.1.1 Cache oblivious

Pro cache oblivious algoritmus jsou zřejmé dva segmenty, s rozhraním okolo cirka 800^2 prvků. Skok mezi těmito dvěma segmenty není nikterak výrazný, z $1.5*10^{-6}$ na $4.0*10^{-6}$.

Tento skok přibližně odpovídá tomu, kdy se celá matice (640000 prvků, 3MB) ještě vleze do L3 cache. Skok tedy začíná trochu dříve než by čistě matematicky měl. I pro $n == 1000$ (4MB) by se do L3 matice měla vejít celá, nicméně kvůli asociativitě cache a komplikovanosti dnešního HW není úplně překvapivé, že ke zhoršování průměrného času dochází před úplným naplněním.

Byť to z grafu není úplně vidět, tak čistá data naznačují, že podobný skok by mohl být i u hranice velikosti L2 a L3 cache. Konkrétně pro $n \approx 256$, kdy se celá matice ještě vejde do L2, se průměrný čas přístupu zvedne z $1.2 * 10^{-6}$ na $1.4 * 10^{-6}$. Kvůli šumu a skokům k vyšším hodnotám pro určitá n (téměř jistě kvůli asociativitě) se ovšem nedá říct nic s jistotou.

Na rozhraní n pro matice, které se těsně vejdou do L1 cache žádný pozorovatelný rozdíl není.

0.1.2 Naive

U naivního algoritmu jsou jasně pozorovatelné segmenty 3. Od začátku do 1000^2 prvků, kde se průměrný čas pohybuje okolo $1.5*10^{-6}$, tedy stejně jako u cache oblivious algoritmu. Následně do 20000^2 prvků, kdy je čas okolo $1 - 2 * 10^{-5}$, a nakonec poslední segment, ve kterém čas roste až k $5 * 10^{-5}$ milisekund na prohození.

První skok odpovídá již dříve popsanému přechodu z toho, kdy se celá matice ještě vleze do L3 cache a kdy už ne. Dokud se vleze, tak je naivní algoritmus, s výjimkou $n == 512$, naprosto srovnatelný s cache aware variantou. Pro $n == 512$ se téměř určitě projevuje asociativita cache. Ta má která má, vzhledem k access patternu, daleko větší dopad v případě naivního přístupu (kde přistupujeme postupě na všechny řádky pod sebou, které jsou od sebe přesně o $4 * 512$B) než v případě cache oblivious.

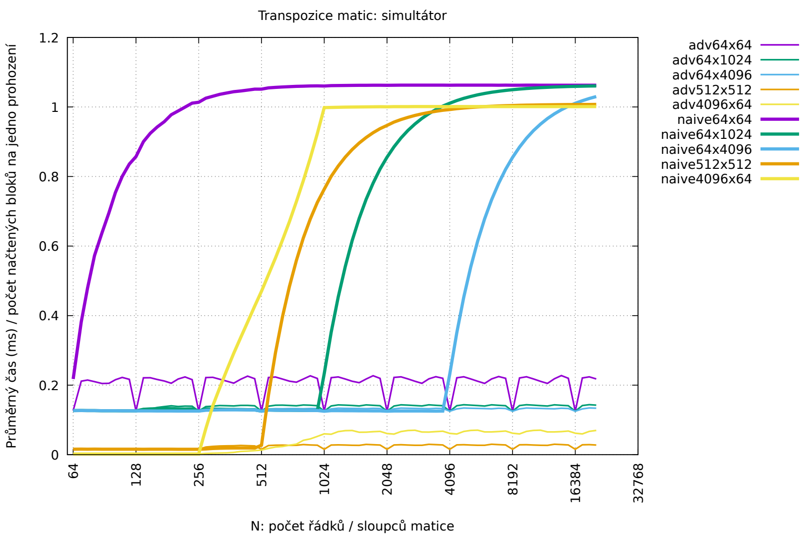
To že je naivní přístup až do n, kdy se matice nevejde do L3 cache, srovnatelný s cache oblivious algoritmem jen dále podporuje tezi, že mezi tím, jestli se matice vejde do L1, L2 nebo jen L3 není žádný významně pozorovatelný rozdíl, který by nebyl z větší části vynahrazen cache pre-fetchem a zamaskován ostatními vlivy. Pokud by totiž přesouvání z L3 cache do L1 a L2 mělo výrazný overhead, tak by se tento overhead nutně více projevil v naivním algoritmu (cache oblivious přístup tyto přesuny minimalizuje).

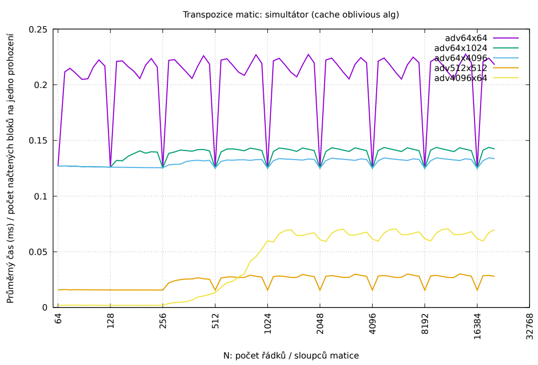
Od n, kdy se matice už nevleze do L3 cache roste průměrná doba swapu postupně až do $n \approx 22000$ (matice velikosti 2GB), kdy začne raketově růst z $2 * 10^{-5}$ k $5 * 10^{-5}$ milisekundám. Tam se následně, byť v velkým rozptylem, drží až do maximální testované velikostí $n == 44591$ (7.9 GB).

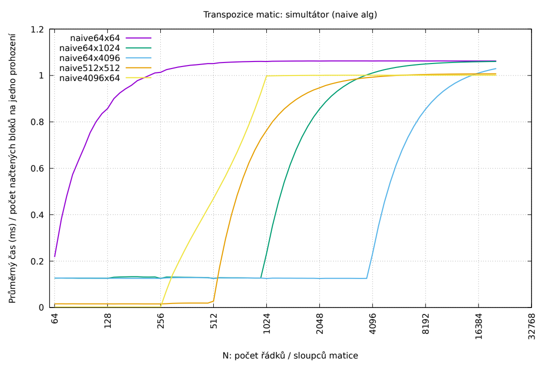
Tento růst a obecně velký rozptyl v druhém i třetím segmentu si vysvětlují tím, jak je paměť RAM strukturovaná. Za zmínku stojí ještě vyšší průměrný čas vždy když je n násobkem mocin 2. Z části to bude způsobené tím, že u poloviny přístupů (ty co jdou v rámci řádku) se bude více projevovat cache aliasing (viz výše) způsobený asociativností cache. Z části to může být způsobené také strukturou paměti RAM, nicméně na posouzení toho co má konkrétně jaký vliv nemám dostatečné znalosti.

0.2 Simulátor

Na grafu ze simulátoru jsou


Transpozice matic: simultátor
Průměrný čas (ms) / počet načtených bloků na jedno prohození
N: počet řádků / sloupců matice
1.2
1
0.8
0.6
0.4
0.2
0
64
128
256
512
1024
2048
4096
8192
16384
32768
adv64x64
adv64x1024
adv64x4096
adv512x512
adv4096x64
naive64x64
naive64x1024
naive64x4096
naive512x512
naive4096x64


Transpozice matic: simultátor (cache oblivious alg)
Průměrný čas (ms) / počet načtených bloků na jedno prohození
N: počet řádků / sloupců matice
0.25
0.2
0.15
0.1
0.05
0
64
128
256
512
1024
2048
4096
8192
16384
32768
adv64x64
adv64x1024
adv64x4096
adv512x512
adv4096x64


Transpozice matic: simultátor (naive alg)
Průměrný čas (ms) / počet načtených bloků na jedno prohození
N: počet řádků / sloupců matice
1.2
1
0.8
0.6
0.4
0.2
0
64
128
256
512
1024
2048
4096
8192
16384
32768
naive64x64
naive64x1024
naive64x4096
naive512x512
naive4096x64