Posudek na diplomovou práci

# Cache-oblivious Algorithms

Michal Vaner

Práce se zabývá cache-oblivious algoritmy, jejich analýzou a chováním v reálných podmínkách moderních počítačů.

V prvních třech kapitolách práce student definuje termíny používané ve zbytku práce a popisuje různé výpočetní modely, které souvisí s tématem práce.

Ve čtvrté kapitole je popsána metodika měření charakteristik testovaných algoritmů. K této metodice mám drobné výhrady – cílem práce je měřit chování algoritmů na reálném hardware, přesto student vytváří umělé testovací prostředí, ve kterém pomocí simulace měří počet cache missů a výpadků stránek. Moderní procesory a operační systémy přitom poskytují techniky, které umožňují tyto charakteristiky měřit přímo za běhu programu. Myslím, že takové výsledky by měly větší vypovídací hodnotu (braly by v úvahu reálné chování cache, její asociativitu, lokální proměnné apod.), a tedy byly by cennější. Dále není uvedena velikost operační paměti a velikost L1 a L2 cache počítače, na kterém měření probíhala, což vzhledem k tématu práce považuji za důležité.

V páté kapitole je uveden souhrn nejčastěji užívaných technik při návrhu cache-oblivious algoritmů.

Šestá kapitola se zabývá třídicími algoritmy. Srovnává klasické algoritmy (quick-sort, heap-sort, merge-sort) a jejich varianty s variantami cache-oblivious algoritmu funnel-sort. V měření, které zachycuje celkový čas běhu algoritmu, je funnel-sort pro všechny velikosti měřených vstupních dat poražen algoritmem quick-sort. To je v rozporu s mými zkušenostmi, neboť dle vlastních měření veřejně dostupné implementace algoritmu funnel-sort (The Funnelsort project) je za stejných podmínek jako měření v této práci quick-sort o 10-20% pomalejší.

Sedmá kapitola porovnává různé algoritmy pro násobení matic a různé reprezentace matic v paměti s ohledem na cache.

Následující kapitola popisuje cache-oblivious datové struktury (Buffer repository tree a Buffer priority tree) používané v cache-oblivious grafových algoritmech a algoritmy pro průchod grafem do šířky a do hloubky. Z měření vyplývá, že rychlost cache-oblivious algoritmů je výrazně nižší než rychlost klasických algoritmů. K uvedeným grafům bych měl drobnou výtku – z popisku „Number of elements“ není zjevné, zda se jedná o počet vrcholů grafu nebo jeho hran.

Devátá kapitola popisuje algoritmy pro hledání komponent souvislosti. Student zde popisuje vlastní algoritmus, který tento problém řeší metodou rozděl a panuj. Tento algoritmus je zajímavý z teoretického hlediska, ale dle výsledků není pro praktické použití příliš vhodný, i když v určitých případech je rychlejší než jiné cache-oblivious algoritmy. Obrázek 9.6 bohužel nemá velkou vypovídací hodnotu, neboť i pro největší vstupy je velikost cache (32MB) srovnatelná s velikostí vstupních dat. To platí v určité míře i pro obrázky 9.4 a 9.5.

Kapitola deset se zabývá algoritmy pro hledání množinově největšího párování. I zde student uvádí vlastní algoritmus založený na metodě rozděl a panuj, který ale ve většině případů bohužel nedosahuje dobrých výsledků.

Následuje shrnutí práce a příloha s množstvím dodatečných měření, která se do práce nevešla.

Práce komplexně a srozumitelně pokrývá problematiku základních cache-oblivious algoritmů. Bohužel ale chybí jakékoliv porovnání cache-oblivious algoritmů s cache-aware algoritmy, což bylo součástí zadání práce. Negativně hodnotím i to, že pouze čas běhu algoritmu vychází z reálného spuštění algoritmu na počítači, zatímco ostatní výsledky jsou odvozeny na základě simulací cache (i když použitá metodika je technicky vyřešená pěkně). Navíc jsou některá měření dle mého názoru provedená na příliš malých datech na to, aby byla relevantní. Jedná se hlavně o některá měření na grafech typu sparse, tree-like a triangulation. Navíc občas není jasné, za jakých podmínek byla provedena, např. u A.14 - A.16 není uvedena velikost vstupního grafu.

Výtku mám i proti nesystematičnosti parametrů některých měření. Např. počítač, na kterém se měřil čas vykonávání, měl L3 cache 6MB, počet cache missů u měření 6.2 byl spočítán pro 8MB, u 7.4 pro 1MB a u 9.5 je velikost cache-line zvolena na 1024B (oproti 4096B použité v 9.6), u ostatních měření počtu cache-missů není velikost cache-line uvedena. To bohužel způsobuje, že se nedá zkoumat souvislost mezi počtem cache-missů a časem běhu, neboť tyto údaje nebyly naměřeny za stejných podmínek.

Kladně hodnotím uvedení vlastních algoritmů v kapitolách 9 a 10, které sice v praxi nedopadly nejlépe, ale z teoretického hlediska jsou zajímavé. Rovněž pozitivně hodnotím kvalitu zpracování teoretické části práce.

Celá práce je zpracována kvalitně, je napsána srozumitelně, bez větších stylistických, gramatických či typografických chyb, i když drobnou výtku bych měl k tomu, že číslování stránek v obsahu neodpovídá skutečnosti. Strukturální členění práce je rovněž v pořádku.

Ačkoliv jsem zde uvedl řadu výtek, které mám k této práci, žádnou z nich nepovažuji za zásadní, a proto doporučuji tuto diplomovou práci k obhajobě.

Dne 7.5.2012 v Praze

Mgr. Zbyněk Falt