Úvod

Pri tvorbe efektívnych algoritmov sa najčastejšie zaujímame o časovú zložitosť a prípadne aj pamäťovú zložitosť. To dosahujeme analýzou v takzvanom *RAM* modeli, v ktorom je prístup k dátam v pamäti považovaný za operáciu vykonateľnú v konštantnom čase. V prípadoch, keď pracujeme s veľkým objemom dát, ktoré sú uložené na médiu s veľkou kapacitou ale nízkou prístupovou rýchlosťou, však tento model prestáva vystihovať realitu.

V skutočnosti však nemusí ísť priamo o veľmi veľké objemy dát a pomalé médiá. Rozdiely v prístupových rýchlostiach medzi vyrovnávacou pamäťou na procesore a hlavnou operačnou pamäťou sú dostatočne veľké na to, aby sa nám v praxi oplatilo minimalizovať počet presunov aj medzi nimi. Ich veľkosť je pritom rádovo len niekoľko kilobajtov, čo je častokrát podstatne menej ako objem spracúvaných dát a nemôžeme teda predpokladať, že k týmto pomalým prístupom nedôjde.

Dôležitým faktorom teda môže byť aj počet týchto pamäťových operácií. Tie majú priamy vplyv na výslednú časovú zložitosť, a preto je výhodné minimalizovať množstvo zápisov a čítaní z pomalého úložiska a snažiť sa využívať práve rýchlu vyrovnávaciu pamäť.

Klasickým prístupom v týchto prípadoch sú takzvané *cache-aware* algoritmy, ktoré poznajú presné parametre pamäťového systému pre dosiahnutie požadovanej efektivity. Následkom toho môže byť viazanosť na konkrétny systém či náročnosť a komplikovanosť implementácie. V tejto práci sa budeme venovať *cache-oblivious* algoritmom a dátovým štruktúram, ktoré tieto parametre nepoznajú, no napriek tomu sú v istých prípadoch asymptoticky rovnako efektívne ako ich *cache-aware* ekvivalenty.

Okrem samozrejmej výhody, kedy nám stačí jedna univerzálna implementácia algoritmu pre ľubovolné množstvo rôznych systémov, prinášajú tieto algoritmy aj iné zlepšenia. V takmer každom systéme je pamäť hierarchická – zložená z viacerých úrovní, pričom každá je väčšia ale pomalšia ako tá predošlá. Pri *cache-aware* by pre optimálne využitie každej z úrovní pamäte bolo potrebné poznať parametre každej úrovne, a rekurzívne vnoriť do seba mnoho inštancií, každú optimalizovanú pre jednu úroveň. No v prípade *cache-oblivious* algoritmov a dátových štruktúr nám stačí jedna inštancia - keďže nevyžaduje poznať parametre žiadnej úrovne, bude rovnako efektívna bez ohľadu na ich hodnoty a teda rovnako efektívna na každej úrovni súčasne.

V tejto práci vysvetľujeme problematiku analýzy počtu pamäťových operácií, popisujeme *cache-oblivious* pamäťový model, porovnávame ho s *cache-aware* modelom a uvádzame prehľad vybraných algoritmov a dátových štruktúr. Pre popísané štruktúry uvádzame analýzu ich správania v *cache-oblivious* modeli.

Hlavným výsledkom práce je implementácia vizualizácií týchto dátových štruktúr. Vizualizácie sú implementované ako rozšírenie programu *Gnarley trees*, ktorý vznikol ako bakalárska práca Jakuba Kováča a neskôr bol rozšírený o ďalšiu funkcionalitu v ročníkových projektoch a bakalárskych prácach. Tento program sme rozšírili o podporu pre simuláciu vyrovnávacej pamäte a pridali sme vizualizácie niekoľkých *cache-oblivious* dátových štruktúr. Cieľom je umožniť užívateľom experimentovať s týmito štruktúrami, sledovať ich správanie krok po kroku. Tieto kroky sú podrobne vysvetlené, čo uľahčuje porozumeniu ich fungovania.

Literatúra

[1] Aggarwal, Alok ; Vitter, Jeffrey u. a.: The input/output complexity of sorting and related problems. In: *Communications of the ACM* 31 (1988), Nr. 9, S. 1116–1127

[2] Bayer, Rudolf: Binary B-trees for Virtual Memory. In: *Proceedings of the 1971 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*. New York, NY, USA : ACM, 1971 (SIGFIDET '71), 219–235

[3] Bender, Michael A. ; Demaine, Erik D. ; Farach-Colton, Martin: Cache-Oblivious B-Trees. In: *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2000)*. Redondo Beach, California, November 12–14 2000, S. 399–409

[4] Bender, Michael A. ; Demaine, Erik D. ; Farach-Colton, Martin: Cache-Oblivious B-Trees. In: *SIAM Journal on Computing* 35 (2005), Nr. 2, S. 341–358

[5] Bender, Michael A. ; Duan, Ziyang ; Iacono, John ; Wu, Jing: A locality-preserving cache-oblivious dynamic dictionary. In: *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms* Society for Industrial and Applied Mathematics, 2002, S. 29–38

[6] Demaine, Erik D.: Cache-Oblivious Algorithms and Data Structures. In: *Lecture Notes from the EEF Summer School on Massive Data Sets*. BRICS, University of Aarhus, Denmark, June 27–July 1 2002

[7] Drepper, Ulrich: What every programmer should know about memory. In: *Red Hat, Inc* 11 (2007)

[8] Frigo, Matteo ; Leiserson, Charles E. ; Prokop, Harald ; Ramachandran, Sridhar: Cache-oblivious algorithms. In: *Foundations of Computer Science, 1999. 40th Annual Symposium on* IEEE, 1999, S. 285–297

[9] Intel Corporation: *Intel® 64 and IA-32 Architectures Optimization Reference Manual*. 2014 (248966-029)

[10] Intel Corporation: *Intel® 64 and IA-32 Architectures Software Developer's Manual*. 2014 (325462-050US)

[11] Kasheff, Zardosht: *Cache-oblivious dynamic search trees*, Massachusetts Institute of Technology, Diss., 2004

[12] Kotrlová, Katarína: *Vizualizácia háld a intervalových stromov*, Univerzita Komenského v Bratislave, bakalárska práca, 2012

[13] Kováč, Jakub: *Vyhľadávacie stromy a ich vizualizácia*, Univerzita Komenského v Bratislave, bakalárska práca, 2007

[14] Lukča, Pavol: *Perzistentné dátové štruktúry a ich vizualizácia*, Univerzita Komenského v Bratislave, bakalárska práca, 2013

[15] Prokop, Harald: *Cache-oblivious algorithms*, Massachusetts Institute of Technology, Diss., 1999

[16] Tomkovič, Viktor: *Vizualizácia stromových dátových štruktúr*, Univerzita Komenského v Bratislave, bakalárska práca, 2012