Eötvös Loránd Tudományegyetem

Informatikai Kar

Algoritmusok és Alkalmazásaik Tanszék

Cache-optimális algoritmusok elemzése

Szabó László Ferenc
Habilitált egyetemi docens

Nagy Péter
Programtervező Informatikus MSc

Budapest, 2022

Tartalomjegyzék

1. fejezet

Bevezetés

Yada-yada, ma már kenyérpirítókba is olyan számítógépet tesznek, aminek több memóriaszintje van. Valamint kíváncsi vagyok, hogy a szakdolgozathoz megálmodott szita mennyire cache-oblivious.

2. fejezet

Cache modellje

- external memory model
- két féle műveletigény: lépések száma vs memóriaműveletek száma
- cache aware algoritmus
- cache replacement policy, MIN, LRU, FIFO
- cache oblivious algoritmus
- cache miss okai

2.1. MIN

MIN avagy OPT policy-t ismerjük off-line esetben. MIN és LRU is rendelkezik azzal a tulajdonsággal, hogy egy nagyobb cache minden elemet tartalmaz, amit egy kisebb cache tartalmazna, egy adott műveletsorozaton.

2.2. LRU lemma

Nem túl szigorú feltételek mellett LRU aszimptotikusan ugyanolyan jó, mint a MIN. Elemzésnél választhatunk, hogy melyiket használjuk.

3. fejezet

Cache oblivious agoritmusok

3.1. Lista

Ha a memóriában nem túl sok egybefűggő részen vannak sorban a lista elemei, akkor az cache-oblivious végigolvasni.

3.2. Rendezés

Oszd meg és uralkodj. 2 algoritmus is van. Az alsó korlát elérhető, de a papír nem az igazi erről. Tall cache assumption. A tall cache assumption lényegesen befolyásolja a műveletigényt.

3.3. Mátrix szorzás

Oszd meg és uralkodj.

3.4. FFT

Oszd meg és uralkodj. Alsó korlát is van, amit elér, de túl erős feltevésekkel.

3.5. Diff.egyenlet közelítés

Oszd meg és uralkodj. Több dimenzióban.

3.6. Statikus keresőfa

Van Emde Boas fák. Dinamikus eset szuperbonyolult, asszem.

3.7. Szakdolgozat-szita, aminek nincs rendes neve

A műveletigénynél kicsit majd erőszakoskodni kell tételekkel, amik a végtelenben igazak.

3.8. Párhuzamos algoritmusok (nem lesz)

Végül nem akarok velük foglalkozni. Van egy mendemonda, hogy cache oblivious algoritmusok aránylag jól tűrik, hogy más programokkal együtt futnak. Valamint a work stealing scheduler jól működik együtt a cache-sel, szintén csak mese.

4. fejezet

Szimulátor

4.1. Igazi cache

Asszociativitás. Több szint. Több mag osztozik. Léteznek egészen speciális cache-ek, TLB, decoded microcode, trace cache, ...

4.2. Megoldás

C egy nagyon szűk része, csak annyi, hogy nagy int és float tömböket rekurzív függyvényekkel módosít. Nincs dinamikus allokáció. Nincs párhuzamosság. Nincs I/O. Nincs semmi op.rendszer.

Igazi gépen a kedvenc C fordítóval futtatható, és lehet méricskélni cache-grinddal, vagy hardver számlálókkal.

Szimulálásnál valami saját belső gép utasításaira fordítja, ezt a gépet szimulálja, és minden memóriaműveletet naplóz. Utólag valamilyen cache-hierarchián meg lehet határozni a futási sebességet. (Ha csinálnék párhuzamos programokat, akkor nem lehet utólag sebességet rendelni a futáshoz, mert az utasítások sorrendjét befolyásolhatja a cache-re várakozás.) A programot is memóriában kell tárolni.

4.3. Választható cache-ek

- utasítás/adat/mindkettő
- MIN
- LRU
- FIFO, LIFO, LFU, ..., kevésbé izgalmasak
- teljes/1-2-3...-n asszociativitás

- mindenféle méretben
- egymás után akár több is

5. fejezet

Irodalomjegyzék

[1] A. O. L. Atkin, D. J. Bernstein: Prime sieves using binary quadratic forms, Mathematics of Computation, Volume 73 (2004) 1023–1030