Áttekintés

- A félév során bekapcsolódtam egy projektbe, amely P4 programok költségbecslésével foglalkozott. [3]
- A P4 programokat hálózati csomagok feldolgozására használják, és fontos kérdés, hogy a csomagok feldolgozása milyen sebességgel és erőforrásigénnyel történik.
- Én ezen belül azt vizsgáltam, hogy a programban a keresőtáblák használata mekkor memóriakésleltetéssel jár bizonyos gyorsítótárak használatával.

P4

Programming Protocoll-independent Packet Processors [4]

- IPv4 és IPv6 csomagok irányítása
- IP alapján MAC cím kiválasztása
- MAC cím alapján fizikai port-ra küldés

Keresőtáblák [5]

- excact (*)
- ternary
- lpm

Gyorsítótárak

- Többszintű hiearachia
- Inkluzív
- ARM: (pseudo)random cache eviction (*) [2]
- Cortex-A76 architektúra paraméterei [1]

Hely	Méret	Elérés
L1	64KB	4 ciklus
L2	512KB	11 ciklus
L3	2MB	11 ciklus + 10ns $\approx$ 30 ciklus
RAM	nagy	11 ciklus + 140ns $\approx$ 300 ciklus

Hash tábla

- Kulcsok alapján lehet keresni.
- Ütközés esetén elkezdi lineárisan keresni [5](*)

Jelölések:

- p_s: ugyanazt a mezőt keresi két egymás utáni packet.
- α: használt/teljes méret aránya
- s_l: cacheline mérete
- s_k: mező mérete
- $s_{c(1,2,3,4)}$: cahce mérete
- t_1: L1 cache elérési ideje
- t_{21}: L2 $\rightarrow$ L1 betöltési idő $(t_2 - t_1)$

Hash tábla: használt cahceline-ok száma

Vizsgáljuk meg egy lekérdezés várhatóan hány k_q kulcsot néz meg:

$$k_q = 1 + \mathbb{P}(\text{Foglalt}) * k_q$$

$$k_q = \frac{1}{1-\alpha}$$

Jelölje k a várhatóan használt cacheline-ok számát:

$$k = 1 + k_q \cdot \frac{s_k}{s_l}$$

$$k = 1 + \frac{1}{1-\alpha} \cdot \frac{s_k}{s_l}$$

Valós esetben ez az érték várhatóan kettő alatt lesz.

Hash tábla: elérési idők

Jelölje t_{lc} a cacheline-ok betöltéséből származó időt.
Ha $s_t < s_c$ akkor 0. Különben:

$$t_{lc} = k \left(1 - \frac{s_t}{s_c}\right) (1 - p_s) \cdot t_{21}$$

$$t_{lc} = \left(1 + \frac{1}{1-\alpha} \cdot \frac{s_k}{s_l}\right) \left(1 - \frac{s_c}{s_t}\right) (1 - p_s) \cdot t_{21}$$

Jelölje t_{lk} a mezők végignézésből származó időt az L1-be való beöltés után.

$$t_{lk} = k_q \cdot t_1$$

$$t_{lk} = \frac{1}{1-\alpha} \cdot t_1$$

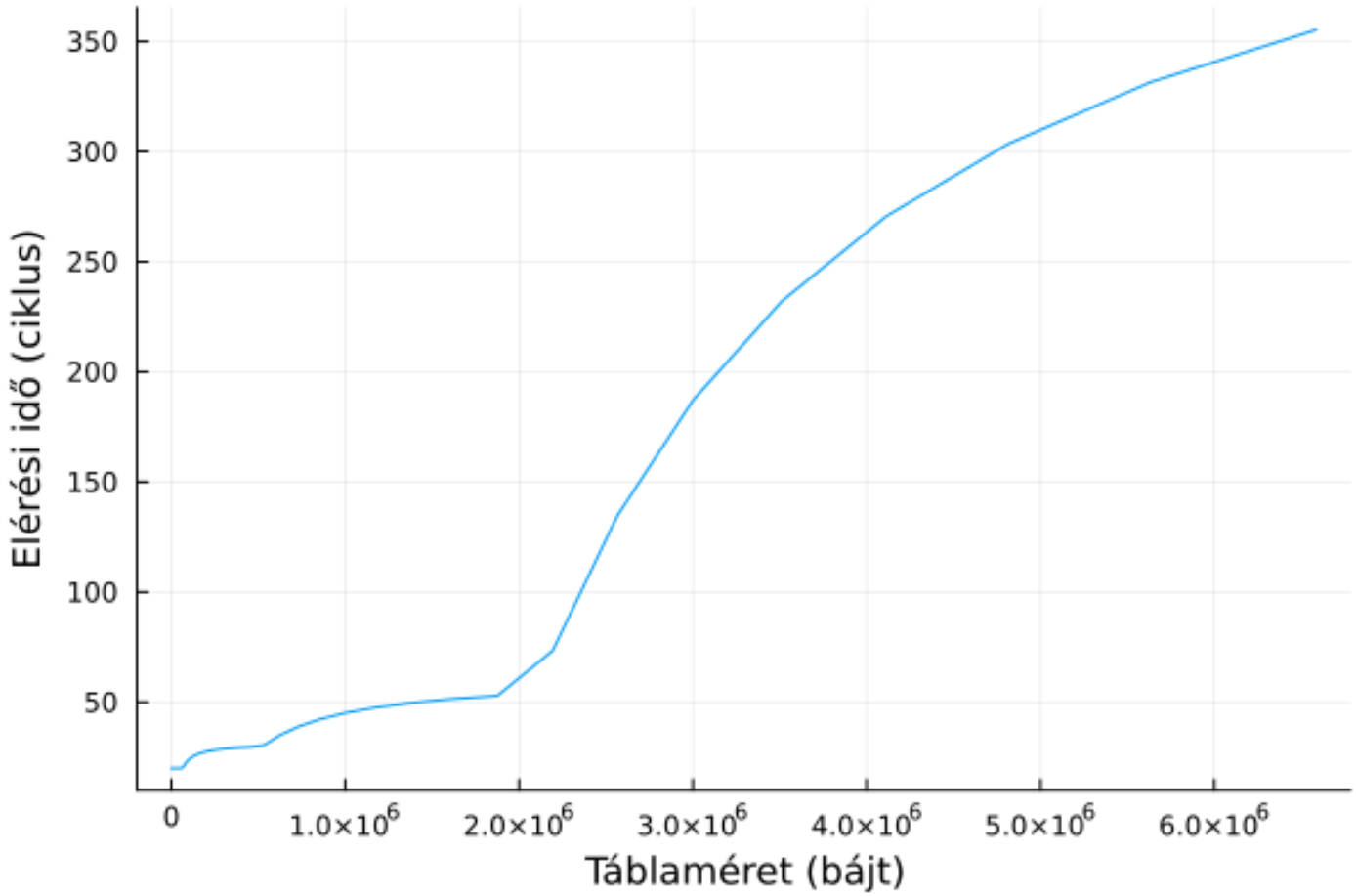
Hash tábla: teljes elérési idő

Többszintű cache esetén $t_{lc1}, t_{lc2}, \ldots$ felírható, az L2 $\rightarrow$ L1 betöltéssel eltöltött idő, L3 $\rightarrow$ L2 betöltési idő.

$$
\begin{aligned}
t_l =& t_{lk} + t_{lc1} + t_{lc2} \\
t_l =& \frac{1}{1-\alpha} \cdot t_1 + (1 - p_s) \cdot \left(1 + \frac{1}{1-\alpha} \cdot \frac{s_k}{s_l}\right) \cdot \\
& \cdot \left(\left(1 - \frac{s_{c1}}{s_t}\right) \cdot t_{21} + \left(1 - \frac{s_{c2}}{s_t}\right) \cdot t_{32}\right)
\end{aligned}
$$

Hash tábla: plot

Cortex-A76 paraméterek, $\alpha = 80\%$, $p_s = 1\%$



Két hash tábla

Két hash tábla verseng a cache-ben maradásért.

- Nevezzük az egyik táblát A-nak, a másikat B-nek.
- s_{cA}: Az A táblának a cache-ben lévő részének a mérete

$$k_A \cdot \mathbb{P}(\text{A érkezik})\mathbb{P}(\text{B kikerül}) = k_b \cdot \mathbb{P}(\text{B érkezik})\mathbb{P}(\text{A kikerül})$$

$$k_A \left(1 - \frac{s_{cA}}{s_{tA}}\right)\left(\frac{s_{cB}}{s_{cA} + s_{cB}}\right) = k_B \left(1 - \frac{s_{cB}}{s_{tB}}\right)\left(\frac{s_{cA}}{s_{cA} + s_{cB}}\right)$$

Ha nincs más a cache-ben, akkor $s_{cA} + s_{cB} = s_c$.

Két hash tábla: cache-ben lévő rész

$$s_{cA} = \frac{s_c(s_{tA} - s_{tB}) - 2s_{tA}s_{tB} \pm \sqrt{s_c^2(s_{tA} - s_{tB})^2 + 4s_{tA}^2 s_{tB}^2}}{2(s_{tA} - s_{t_B})}$$

Ennek az egyenletnek a pozitív megoldását kell választani.

Referenciák

[1] URL: https://www.7-cpu.com/cpu/Cortex-A76.html.

[2] *Data side memory system*. URL: https://developer.arm.com/documentation/ddi0500/j/Functional-Description/About-the-Cortex-A53-processor-functions/Data-side-memory-system?lang=en.

[3] Dániel Lukács, Gergely Pongrácz és Máté Tejfel. "Control flow based cost analysis for P4". *Open Computer Science* 11.1 (2021), 70–79. old. DOI: doi:10.1515/comp-2020-0131. URL: https://doi.org/10.1515/comp-2020-0131.

[4] *P4 wboldala*. URL: https://opennetworking.org/p4/.

[5] *Size property of P4 tables and parser value sets*. URL: https://github.com/p4lang/p4-spec/blob/main/p4-16/spec/docs/p4-table-and-parser-value-set-sizes.md.

Köszönöm a figyelmet.

Eredmények

Egy hash tábla ideje:

$$
\begin{aligned}
t_l =& t_{lk} + t_{lc1} + t_{lc2} \\
t_l =& \frac{1}{1-\alpha} \cdot t_1 + (1 - p_s) \cdot \left(1 + \frac{1}{1-\alpha} \cdot \frac{s_k}{s_l}\right) \cdot \\
& \cdot \left(\left(1 - \frac{s_{c1}}{s_t}\right) \cdot t_{21} + \left(1 - \frac{s_{c2}}{s_t}\right) \cdot t_{32}\right)
\end{aligned}
$$

Két hash tábla közül az egyik cache-ben elfoglalt helye:

$$
s_{cA} = \frac{s_c(s_{tA} - s_{tB}) - 2s_{tA}s_{tB} \pm \sqrt{s_c^2(s_{tA} - s_{tB})^2 + 4s_{tA}^2 s_{tB}^2}}{2(s_{tA} - s_{t_B})}
$$