Návrh cachovacích algoritmů
LFU-SS a LRFU-SS

Autor:
Pavel Bžoch
pbzoch@kiv.zcu.cz

Návrh cache algoritmu – LFU-SS (LFU with Server Statistics)

LFU-SS s hity ze serveru pracuje podobně jako klasické LFU. Jsou zde pouze dva hlavní rozdíly:

- Pro počítání počtu hitů se používá desetinné číslo.
- Pro prvotní výpočet hitů se používají statistiky ze serveru.

Při používání cache mohou nastat tyto tři případy –

1) Uživatel chce soubor, který je v cache.
2) Uživatel chce soubor, který v cache není. Jakmile se soubor stáhne, vejde se do cache.
3) Uživatel chce soubor, který v cache není. Jakmile se soubor stáhne, nevejde se do cache.

Add 1)

LFU s hity funguje stejně jako klasické LFU. Při požadavku na soubor, který je v cache, se zvýší počet hitů u souboru o jedničku. Malý přídavek je tu – jakmile počet hitů u některého souboru převýší mez (třeba 15 hitů), tak se u všech souborů sníží počet hitů na polovinu. Tím se zabrání zastarávání souborů v cache (prevence tzv. ageingu).

Add 2)

Soubor se stáhne a má být umístěn do cache. V normálních LFU by dostal počet hitů roven 1 a byl by umístěn do cache. U LFU s hity vypočteme počet podle vzorečku:

$$READ\_HITS_{klient} = \frac{READ\_HITS_{server} - WRITE\_HITS_{server}}{GLOBAL\_HITS_{server}} \cdot GLOBAL\_HITS_{klient} + 1$$

$READ\_HITS_{server}$ a $WRITE\_HITS_{server}$ jsou hodnoty proměnných, které dostaneme s metadaty k každému souboru ze serveru.

$GLOBAL\_HITS_{server}$ je hodnota celkového počtu přístupů u všech souborů na straně serveru. Tuto hodnotu lze získat ze serveru na přání. Není nutné ji získávat ke každému souboru, pouze ji pravidelně aktualizovat (např. 1x za 15min).

$GLOBAL\_HITS_{klient}$ je součet všech hitů u všech souborů na straně klienta.

Nakonec přičteme jedničku, protože soubor je přistupován klientem. Jakmile vypočteme počet hitů, umístíme soubor do cache a můžeme s ním pracovat.

Add 3)

Když dojde k tomu, že musíme nějaký soubor z cache „vyhodit“, vždy vyhazujeme ten, který má nejmenší počet hitů. Soubory vyhazujeme tak dlouho, dokud není místo pro nově příchozí soubor. Jakmile se soubor do cache vejde, postupujeme podobně jako v Add 2).

Příklad použití LFU s hity (na další straně):

1) v cache jsou 4 soubory:
A - počet hitu 8
B - počet hitů 5
C - počet hitů 5
D - počet hitů 2

$GLOBAL\_HITS_{klient} = 8+5+5+2=20$

2) přichází nový soubor do cache:
F, $READ\_HITS_{server} = 592$, $WRITE\_HITS_{server}=5$, $GLOBAL\_HITS_{server} = 11538$

$READ\_HITS_{klient} = ((592-5) / 11538) * 20 + 1 = 2,0175074$

Cache po příchodu souboru F bude vypadat takto:
A - počet hitu 8
B - počet hitů 5
C - počet hitů 5
F - počet hitů 2,0175074
D - počet hitů 2

3)Soubor F bude znovu přistupován:
F - počet hitů + 1 = 2,0175074 + 1 = 3,0175074

Soubory v cache podle počtu hitů:
A - počet hitu 8
B - počet hitů 5
C - počet hitů 5
F - počet hitů 3,0175074
D - počet hitů 2

Pokud by byl požadavek na vyhození souboru, byl by vyhozen soubor D.

Návrh cache algoritmu – LRFU-SS (hybrid mezi LFU-SS a LRU)

Z obou algoritmů, LFU-SS a LRU, se pokusíme získat 16-bitové celé nezáporné číslo, které bude charakterizovat, jak moc algoritmus „přeje“ danému souboru. Výsledné číslo pro cache pak bude součtem těchto dvou čísel, přičemž použijeme koeficienty, pomocí nichž můžeme upřednostnit jeden či druhý algoritmus. U obou algoritmů bude číslo 0 znamenat nejhorší prioritu, číslo 65535 (2^16 - 1) znamená nejvyšší prioritu.

Zapsáno pomocí vzorečku:

$$P_{výsledná} = K_1 \cdot P_{LRU} + K_2 \cdot P_{LFU-SS}$$

K_1 je celočíselný koeficient pro ovlivnění priority LRU.
P_{LRU} je vypočtená priorita pomocí algoritmu Least Recently Used.
K_2 je celočíselný koeficient pro ovlivnění priority LFU-SS.
P_{LFU-SS} je vypočtená priorita pomocí algoritmu Least Frequently Used with Server Statistics.

Výslednou prioritu je potřeba přepočítávat pokaždé, když dojde k přistupování souboru v cache nebo když přijde nový soubor do cache, protože se priorita mění u všech souborů.

Výpočet priority LRU (P_{LRU})

U každého souboru uchováváme informaci, kdy byl naposledy přistupován (časový otisk – např. 64-bitové číslo udávající čas od 1.1.1970 v nanosec). Zároveň systém (cache) uchovává čas nejdéle nepoužívaného souboru. Pro výpočet priority použijeme lineární interpolaci. V tomto případě nejstaršímu souboru bude odpovídat priorita 0, nejnověji přistupovaný soubor bude mít prioritu 65535.

Vyjádřeno pomocí vzorečku:

$$\mathrm{P_{LRU\_SOUBOR}} = \left(\mathrm{T_{SOUBOR}} - \mathrm{T_{NEJSTARŠÍ\_SOUBOR}}\right) \cdot \frac{65535}{\mathrm{T_{NEJNOVĚJŠÍ\_SOUBOR}} - \mathrm{T_{NEJSTARŠÍ\_SOUBOR}}}$$

T_{SOUBOR} je čas, kdy byl naposledy přistupován soubor, u kterého počítáme prioritu.
$T_{NEJSTARŠÍ\_SOUBOR}$ je čas souboru, který nebyl nejdelší dobu přistupován.
$T_{NEJNOVĚJŠÍ\_SOUBOR}$ je čas souboru, který byl přistupován naposledy (nejnověji).

Je vhodné, aby si cache uchovávala $T_{NEJSTARŠÍ\_SOUBOR}$ a $T_{NEJNOVĚJŠÍ\_SOUBOR}$. Urychlí se tím výpočty priorit.

Výpočet priority LFU-SS (P_{LFU-SS})

Výpočet priority pro LFU-SS je podobný jako u LRU. Opět použijeme lineární interpolaci. Jen připomínám, že počet hitů je desetinné číslo a že u nově příchozích souborů se počítá ze statistik ze serveru. Vyjádřeno pomocí vzorečku:

$$P_{LFU-SS,file} = (READ\_HITS_{file,client} - MINIMUM\_HITS_{client}) \cdot \frac{65535}{MAXIMUM\_HITS_{client} - MINIMUM\_HITS_{client}}$$

$READ\_HITS_{file,\ client}$ je hodnota, která udává, kolikrát byl soubor přistupován klientem (zároveň zohledňuje data ze serveru).
$MINIMUM\_HITS_{client}$ je hodnota, která uchovává hodnotu nejmenšího počtu přístupů k souboru. (minimální hodnota READ_HITS, kterou lze v cache najít).

$MAXIMUM\_HITS_{client}$ je hodnota, která uchovává hodnotu největšího počtu přístupů k souboru. (maximální hodnota READ_HITS, kterou lze v cache najít).

Je tu jedna výjimka - pro **nově příchozí** soubor se $P_{LFU\text{-}SS}$ počítá jinak – nově příchozí soubor chceme trochu upřednostnit, takže se priorita pro něj spočte podle vzorečku:

$$P_{LFU-SS} = \frac{READ\_HITS_{server}}{GLOBAL\_HITS_{server}} \cdot 65535$$

Ve vzorečku se používají statistiky, které získáme ze serveru.

Dále bude popis jednotlivých akcí, které mohou nastat pro cache (podobně jako u LFU-SS):

1) Uživatel chce soubor, který je v cache.
2) Uživatel chce soubor, který v cache není. Jakmile se soubor stáhne, vejde se do cache.
3) Uživatel chce soubor, který v cache není. Jakmile se soubor stáhne, nevejde se do cache.

Add 1)

U požadovaného souboru se zvýší počet hitů a nastaví se mu čas, kdy byl soubor požadován. Vypočtou se priority u všech souborů v cache (u všech souborů se změní P_{LRU}, u některých se může změnit i $P_{LFU\text{-}SS}$).

Add2)
U požadovaného souboru se vypočte $P_{LFU\text{-}SS}$ podle vzorečku pro nově příchozí soubory, vypočte se počet hitů (viz LFU-SS), nastaví se čas přístupu, vypočte se celková priorita. Vypočtou se priority u všech ostatních souborů v cache (u všech souborů se změní P_{LRU}, u některých se může změnit i $P_{LFU\text{-}SS}$).

Add3)
Dokud se soubor nevejde do cache, vyhazují se soubory s nejmenší prioritou. Jakmile je uvolněno místo, počítá se stejně jako v add 2).

Příklad hybridního algoritmu (na další straně):

1) V cache jsou 4 soubory (koeficienty $K_1 = 1$, $K_2 = 2$):

A - počet hitů 8, naposledy přistupován 15, $P_{LRU} = 0$; $P_{LFU\text{-}SS} = 65535$; $P_{celková} = 131070$

B - počet hitů 5, naposledy přistupován 48, $P_{LRU} = 45055{,}3125$; $P_{LFU\text{-}SS} = 32767{,}5$; $P_{celková} = 110590{,}3125$

C - počet hitů 5, naposledy přistupován 25, $P_{LRU} = 13653{,}125$; $P_{LFU\text{-}SS} = 32767{,}5$; $P_{celková} = 79188{,}125$

D - počet hitů 2, naposledy přistupován 63, $P_{LRU} = 65535$; $P_{LFU\text{-}SS} = 0$; $P_{celková} = 65535$

Pozn. Pokud by měl být vyhozen soubor, byl by vyhozen soubor D s nejmenší prioritou.

2) Přichází nový soubor do cache:

F, $READ\_HITS_{server} = 592$, $WRITE\_HITS_{server} = 5$, $GLOBAL\_HITS_{server} = 11538$, čas přístupu 75

Pro F: $READ\_HITS_{client} = ((592-5) / 11538) * 20 + 1 = 2{,}0175074$

Cache po příchodu souboru F bude vypadat takto:

A - počet hitů 8, naposledy přistupován 15, $P_{LRU} = 0$; $P_{LFU\text{-}SS} = 65535$; $P_{celková} = 131070$

B - počet hitů 5, naposledy přistupován 48, $P_{LRU} = 36044{,}25$; $P_{LFU\text{-}SS} = 32767{,}5$; $P_{celková} = 101579{,}25$

C - počet hitů 5, naposledy přistupován 25, $P_{LRU} = 10922{,}5$; $P_{LFU\text{-}SS} = 32767{,}5$; $P_{celková} = 76457{,}5$

F - počet hitů 2,0175; naposledy přistupován 75, $P_{LRU} = 65535$; $P_{LFU\text{-}SS} = 3362{,}517$; $P_{celková} = 72260{,}034$

D - počet hitů 2, naposledy přistupován 63, $P_{LRU} = 52428$; $P_{LFU\text{-}SS} = 0$; $P_{celková} = 52428$

Pozn. $P_{LFU\text{-}SS}$ u souboru F (nového) se počítá jinak

Pozn2. Pokud by měl být vyhozen soubor, byl by vyhozen soubor D s nejmenší prioritou

3) Soubor F bude znovu přistupován:

F - počet hitů + 1 = 2,0175074 + 1 = 3,0175074, čas přístupu 95

Soubory v cache podle priority:

A - počet hitů 8, naposledy přistupován 15, $P_{LRU} = 0$; $P_{LFU\text{-}SS} = 65535$; $P_{celková} = 131070$

B - počet hitů 5, naposledy přistupován 48, $P_{LRU} = 27033{,}19$; $P_{LFU\text{-}SS} = 32767{,}5$; $P_{celková} = 92568{,}19$

C - počet hitů 5, naposledy přistupován 25, $P_{LRU} = 8191{,}875$; $P_{LFU\text{-}SS} = 32767{,}5$; $P_{celková} = 73726{,}875$

F - počet hitů 3,02; naposledy přistupován 95, $P_{LRU} = 65535$; $P_{LFU\text{-}SS} = 11113{,}725$; $P_{celková} = 87762{,}45$

D - počet hitů 2, naposledy přistupován 63, $P_{LRU} = 39321$; $P_{LFU\text{-}SS} = 0$; $P_{celková} = 39321$

Pozn. Pokud by byl požadavek na vyhození souboru, byl by vyhozen soubor D.