Labo 2 - Virtual Memory

Boris Michiels, Anthony Tacquet

mei 2025

1 Java code

Onze java-code is opgesplitst in 3 algemene packages: *data*, *globals* en *presentatie*. Al onze constanten die bepalen hoe onze gesimuleerde machine eruitziet, bevinden zich in het *Constants.java* bestand. Onze code werd zo geschreven om makkelijk uitbreidbaar te zijn; ook zou de code moeten werken als andere waarden worden ingesteld in het *Constants.java* bestand voor bijvoorbeeld *page table entries*, *aantal frames in memory* en *address offset bits*.

2 Statistieken

Onze code bevat statistieken voor alle *page ins*, *page outs* en *page evictions* zoals in de opgave gevraagd wordt. In de TUI kan men ook het aantal *page faults* opvragen; het aantal *page faults* zal logischerwijs altijd gelijk zijn aan het aantal *page ins*, een *page fault* zal eerst geregistreerd worden aangezien dit een *page in* actie activeert.

3 Reallocatie-algoritme

3.1 Aged Second Chance (ASC) algoritme

Ons eerste reallocatie-algoritme bestaat uit een combinatie van het *Second Chance* algoritme en het *Least Recently Used (LRU)* algoritme. Het algoritme gaat grotendeels te werk zoals het *Second Chance* algoritme, maar zal als startindex voor het zoeken van een geschikte frame altijd beginnen met itereren bij de frame die al het langste in het geheugen zit. We noemen dit algoritme daarom ook het *Aged Second Chance (ASC)* algoritme.

3.2 Weighted Aged Frequency (WAF) algoritme

In dit (zelf ontworpen) algoritme worden voor elk frame enkele extra waarden bijgehouden:

- *frequency*: aantal keer dat er naar deze pagina gelezen/geschreven wordt;
- *clock*: wanneer deze page voor het eerst in het geheugen werd geladen.

Aan de hand van deze waarden kan het algoritme een verhouding berekenen per frame, en zo een *victim* selecteren. De *victim* die gekozen wordt,

zal vanzelfsprekend uit het geheugen gehaald worden om zo plaats te maken voor een nieuwe pagina. De formule voor het berekenen van de verhouding wordt als volgt bepaald:

$$cost = \frac{age}{frequency + weight \cdot dirty}$$

Age wordt bepaald uit hoe lang een pagina al in het geheugen zit en kan dus bepaald worden door de waarde van de *clock* in een frame af te trekken van de *system clock*.

Weight is een instelbare parameter, hoe hoger men *weight* instelt hoe meer dat de waarde van de *dirty bit* zal doorwegen. Dit betekent dus ook dat pagina's waar naar geschreven is geweest minder vaak zullen uitgeswapt worden (het aantal *page outs* zal verminderen).

3.3 Vergelijking

We hebben een korte vergelijking opgebouwd om duidelijk het verschil te kunnen bepalen tussen ASC en WAF. Voor het WAF-algoritme werd onze experimenteel gevonden optimale weight van **60** ingesteld.

Statistiek	**ASC**	**WAF**
Page Ins	63	63
Page Evictions	27	27
Page Outs	0	0

Tabel 1: Statistieken tussen het ASC en WAF algoritme (few dataset)

Statistiek	**ASC**	**WAF**
Page Ins	5812	5805
Page Evictions	5361	5256
Page Outs	1488	1383

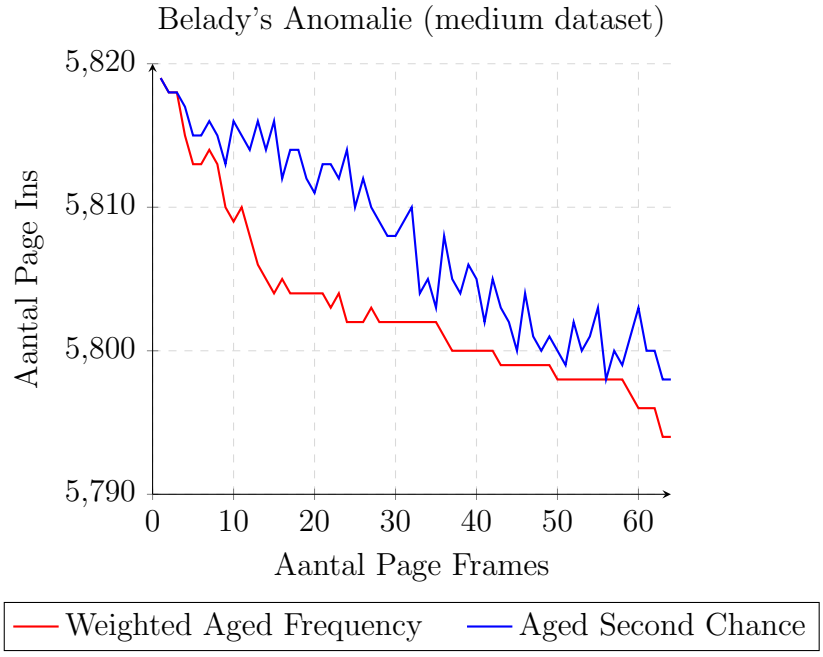
Tabel 2: Statistieken tussen het ASC en WAF algoritme (medium dataset)

Statistiek	ASC	WAF
Page Ins	11374	11372
Page Evictions	11122	11146
Page Outs	3534	3553

Tabel 3: Statistieken tussen het ASC en WAF algoritme (many dataset)

3.4 Belady's Anomalie

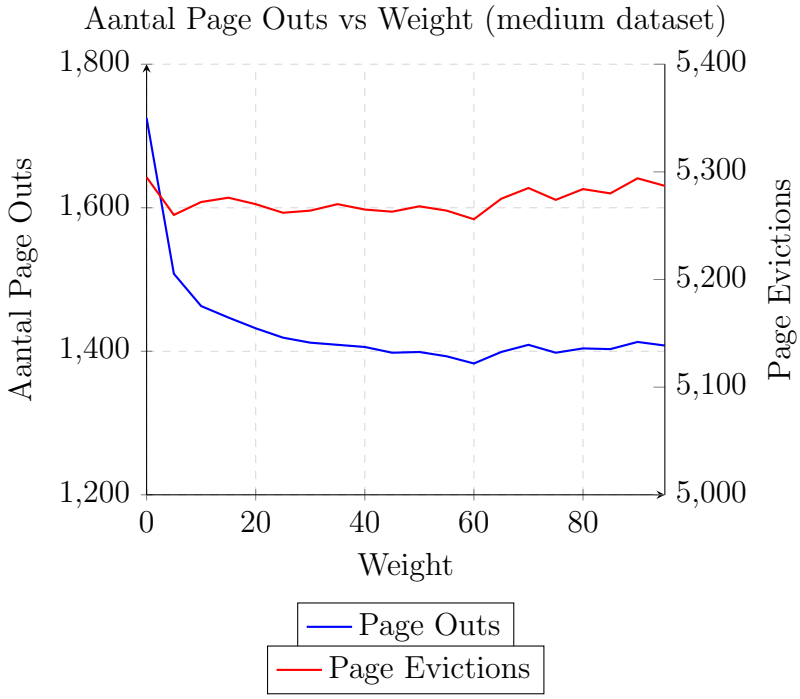
We demonstreerden Belady's anomalie aan de hand van het WAF-algoritme met een *weight* van **35**.



Zoals we gezien hebben in de theorie, zien we dat het algemene verloop van de grafiek neerwaarts is, maar er bevinden zich „sprongen" in de data. Die sprongen verwacht je niet bij het verhogen van het aantal frames.

4 Grafieken

We hebben een grafiek opgesteld om het effect van de *weight*-parameter te visualiseren. De 2D grafiek met als x-as de *weight* en y-as het aantal *page outs* en het aantal *page evictions*, kunnen we zo voorstellen:



Uit de grafiek blijkt dat de optimale *weight* voor de medium dataset, 60 is. Hierbij zijn het aantal *page evictions* en het aantal *page outs* het laagste. Na een bepaald punt kunnen we ook zien dat het verhogen van de *weight*-parameter geen effect meer zal hebben op het aantal *page outs*, maar juist slechter zal presteren.

5 Besluit

Ons project is voorzien van een GUI en een TUI, we hebben een combinatie gebruikt van bestaande reallocatie-algoritmen als inspiratie voor ons eigen reallocatie-algoritme. Ook hebben we een eigen algoritme uitgevonden om het aantal *page outs* te minimaliseren.