Universiteit van Amsterdam

Memory Allocator

Een domme allocator, maar nu iets slimmer

15 mei 2018

Student:
Rick Teuthof
11209119

1 Introductie

Bij het schrijven van een memory allocator moet men rekening houden met een aantal factoren. Een van deze factoren is bijhouden hoe memory gefreed moet worden. Bij een gegeven zogenaamde 'dumb-allocator' is een bijbehorende 'dumb-free' geimplementeerd. In dit rapport wordt beschreven hoe de implementatie van deze free er uit ziet.

2 Methode

Voor de implementatie is de voorbeeldcode van de opdracht volledig overgenomen, en hierbij is de bijbehorende free, die in eerste instantie leeg was, geimplementeerd. Voor het testen van de dumb-free is de implementatie vergeleken met een allocator zonder free.

2.1 Algoritme

Het algoritme van de dumb-free is zeer simpel. Allereerst wordt de pointer naar de geheugenlocatie die gefreed moet worden niet gebruikt. We hebben immers te maken met een dumb-free systeem. de huidige memory bank teller wordt verlaagd met 1, en deze bank wordt vervolgens uitgeschakeld. Op deze manier wordt er wel degelijk memory gefreed, maar dit gaat gepaard met enkele nadelen. Hierover wordt dieper in gegaan in de sectie 'Discussie'.

3 Resultaten

In de volgende tabel worden de simple.t file met 1 module en 42 banken getest voor de allocator met en zonder dumb-free:

Tabel 1: Tests met en zonder free voor 1 module en 42 banken

	Met free	Zonder free
op/s	12.151	18.934
op/J	6.688	6.000

Uit deze tabel is op te maken dat de performance sterk omlaag gaat, maar de stroomefficientie is wel iets verhoogd.

4 Discussie

Bij deze implementatie van free wordt er verwacht dat er geen meerdere allocaties achter elkaar plaats vinden. Stel de volgende situatie: de gebruiker alloceert geheugen voor twee objecten, a en b in die volgorde, als men a nu wilt free'en wordt door de dumb-free implementatie het geheugen van b gefree'd. Dit zou opgelost kunnen worden door een free te implementeren waarbij er een administratie geimplementeerd is die de actieve banks bijhoudt.

Verder is de dumb-free nadelig voor de performance van het programma. Hoewel de stroomefficientie omhoog gaat is dit verschil te laag om een goede vervanging te zijn voor performance.

4.1 Conclusie

Door de sterke nadelen is de dumb-free niet geschikt als een efficiente free implementatie.