Universiteit van Amsterdam

Opgave 2: Scheduler

Verschillende scheduler-algoritmen en hun invloed op de snelheid van geheugentoewijzing en de lengte van uitvoeringstijd

29 april 2018

Student:
R.A.J. Wacanno
11741163

Practicumgroep:
A1 - ALGOL

Docent:
S. Altmeyer

Cursus:
Besturingssystemen

Vakcode:
5062BEST6Y

1 Introductie

Het kiezen van een algoritme voor CPU-*scheduling* kan impact hebben op bepaalde prestatie-eigenschappen van het systeem waarin het geïmplementeerd wordt. Er moeten daarom afwegingen gemaakt worden over welke eigenschappen positief beïnvloed moeten worden in een bepaalde context. In sommige gevallen zou het wenselijk kunnen zijn dat een proces zo snel mogelijk geheugen toegewezen krijgt. Ook zou het zo kunnen zijn dat de gemiddelde uitvoeringstijd zo klein mogelijk moet zijn. Deze vraag naar geheugen-efficiëntie en een korte uitvoeringstijd staan specifiek in dit onderzoek centraal.

2 Methode

2.1 Algoritme

Voor het uitvoeren van het onderzoek moesten drie verschillende *scheduling*-algoritmen geschreven worden. Een simpele, wiens prestaties als basis dienen voor de verdere metingen en verder eentje die geheugen-efficiënt is en eentje die de totale uitvoeringstijd kan verminderen.

Round Robin
Het *Round Robin*-algoritme is als basis geïmplementeerd en zal tevens als ondergrond dienen voor de uitgebreidere algoritmen. Dit algoritme werkt op basis van één enkele rij waarvoor geld dat hetgeen eerst komt er ook weer als eerste uit gaat. Het proces dat vooraan in de rij staat wordt uit de rij gehaald, krijg voor een bepaalde hoeveelheid tijd — een tijdsquantum — de mogelijkheid om op de *CPU* te draaien en — mits hij na deze tijd nog niet klaar is

— wordt achteraan weer terug in de rij geplaatst. Deze cyclus gaat zo door totdat de rij helemaal leeg is.

***Round Robin met* kleine aanpassingen**
Om er voor te zorgen dat processen zo snel mogelijk geheugen toegewezen krijgen, moest aan het bovengenoemde algoritme het een en ander worden aangepast. Ook moest de manier waarop het geheugen toegewezen werd op een andere manier worden geïmplementeerd.

Hierbij ging het vooral om het kiezen van de juiste volgorde. Bij het toewijzen van het geheugen krijgt in deze implementatie steeds het proces met de laagste geheugenbehoefte als eerste geheugen toegewezen. In de *CPU-scheduler* krijgt steeds het proces met de grootste geheugenbehoefte als eerste de beurt.

Op deze manier probeer je het geheugen met zo veel mogelijk verschillende processen te vullen en maak je zo snel mogelijk weer zo veel mogelijk ruimte vrij. Dit kan er voor zorgen dat sommige individuele processen iets langer moeten wachten, maar het zou de gemiddelde wachttijd moeten lagen

Multi Level Feedback Queue
De *Multi Level Feedback Queue* (*MLFQ*) is in feite een meerlaagse *Round Robin*. Het zijn meerdere rijen met ieder een andere prioriteit en ander tijdsquantum.

Processen beginnen in de bovenste rij en uit deze rij worden ook steeds processen gehaald voor uitvoering. Als een proces langer dan een tijdsquantum van de betreffende rij bezig is wordt hij een rij lager teruggeplaatst. De lagere rijen hebben steeds een hoger tijdsquantum. Als een proces wordt gestopt door een behoefte aan in- of uitvoer wordt dit proces een rij naar boven verplaatst. Het uitvoeren van processen begint steeds bij de bovenste rij en, zodra deze leeg is, gaat het uitvoeren verder op één rij lager. Mocht er dan in een hogere rij toch een proces bij komen, dan krijgt dit proces voorrang.

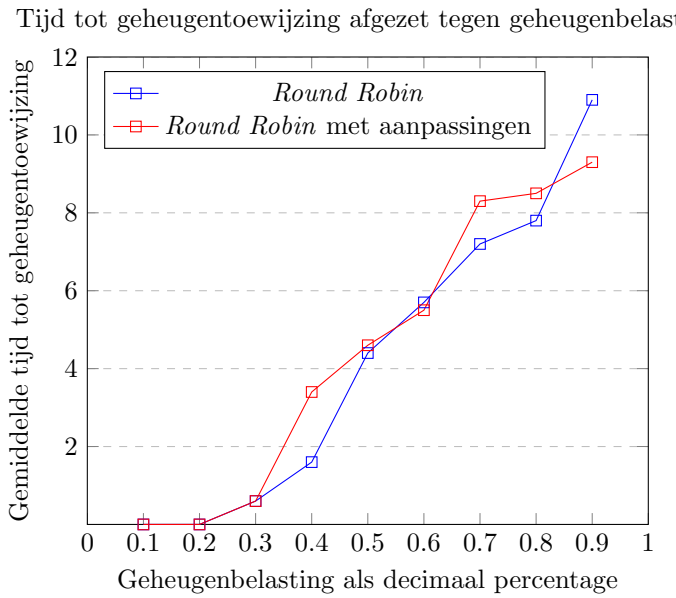
Dit algoritme heeft als gevolg dat kortdurende processen zo snel mogelijk klaar zijn. Langlopende processen krijgen op hun beurt een steeds groter tijdsquantum en dus meer tijd per ronde om uit te voeren. Ook heeft dit algoritme een positieve invloed op processen die in- en uitvoer verwachten.

2.2 Procedure

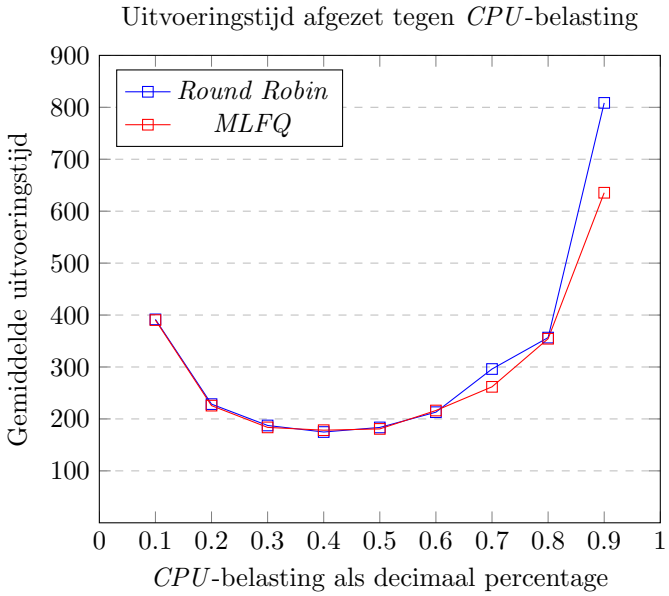
Voor het uitvoeren van het onderzoek worden steeds het de normale *Round Robin* en de veranderde *Round Robin* of *MLFB* met dezelfde startparameters gedraaid. Om geheugen-efficiëntie te testen worden te eerste en tweede met een steeds oplopende (stappen van 0,1) geheugenbelasting gedraaid. Voor het testen van de uitvoeringstijd worden de eerste en derde gedraaid met een steeds oplopende (wederom stappen van 0,1) *CPU*-belasting. Vervolgens worden bij deze twee onderdelen respectievelijk de gemiddelde tijd tot geheugentoewijzing en gemiddelde uitvoeringstijd genoteerd.

3 Resultaten

Hieronder zijn in figuren 1 en 2 de resultaten van het onderzoek weergegeven.



Figuur 1: De gemiddelde tijd tot geheugentoewijzing bij verschillende maten van geheugenbelasting met een in- en uitvoerbelasting van 0,1, een *CPU*-belasting van 0,1 en 10000 processen



Figuur 2: De gemiddelde uitvoeringstijd bij verschillende maten van geheugenbelasting met een in- en uitvoerbelasting van 0,4, een geheugenbelasting van 0,3 en 10000 processen

4 Discussie

Uit de verkregen resultaten komen een aantal dingen naar boven. Allereerst kan gesteld worden dat het geheugen-efficiënte (*Round Robin* met aanpassingen) algoritme niet effectief was en in sommige gevallen zelfs averechts werkt. De effecten van het algoritme werden sterk beïnvloed door de te stellen parameters (in- en uitvoerbelasting en *CPU*-belasting). Enigszins hoopvol is dan nog het feit dat bij de hoogste geheugenbelasting, het algoritme daadwerkelijk ook beter presteert.

De *Multi Level Feedback Queue* lijkt in de eerste instantie ook geen grote verbetering. Diens prestaties blijven rond die van de *Round Robin* hangen en blijken in sommige gevallen zelfs slechter te zijn. Bij hoge CPU-belasting — tussen de 70 en 100 procent — komt hier echter verandering in. De *MLFQ* presteert in deze omstandigheden significant beter.

4.1 Conclusie

Al met al kan word duidelijk dat een scheduling-algoritme impact kan hebben op prestaties wanneer deze op de juiste manier gekozen en geïmplementeerd wordt. De *Round Robin* met aanpassingen om het geheugen-efficiënter te maken was niet effectief en in de meeste gevallen zelfs langzamer dan de normale *Round Robin*.

De *Multi Level Feedback Queue* bleek echter wel een goed instrument om executietijden bij hoge CPU-belasting te verkorten ten opzichte van de normale *Round Robin*.