Firefox isoleren in een Linux-cgroup

Johan Swenker

Dit artikel is geïnspireerd op een probleem van Jan[1], een van de leden van de HCC Linux-werkgroep in Groningen. Zijn Linux-computer crasht op (on)regelmatige tijden. De computer wordt dan door het swappen zodanig onbruikbaar dat er weinig anders op zit dan die maar uit te zetten. Binnen Linux kun je processen van elkaar isoleren met cgroups. Maar hoe werkt dat?

Linux Weekly News

Ik ben al sinds jaren abonnee van **https://lwn.net**. Dit is een webzine dat over Linux schrijft. Abonnees krijgen elke donderdag een nieuwe uitgave. Voor andere mensen komt dezelfde informatie met een week vertraging beschikbaar. Door LWN kende de kreet *cgroup*; ik herinner me dat ik een keer met cgroups geëxperimenteerd heb die het cpu-gebruik van processen beperken. Maar verder heb ik een zoekmachine nodig om me de details te vertellen.

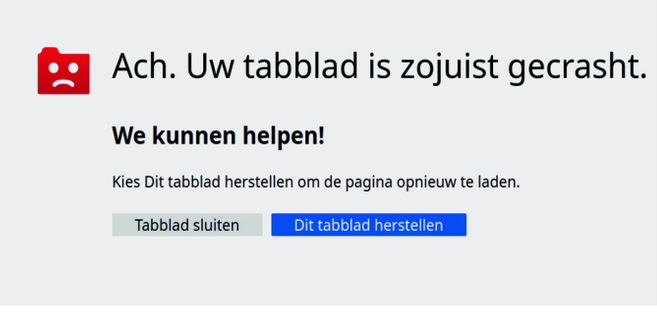
Control group

Het sleutelwoord waar het om gaat is dus control-group (cgroup). Met cgroups kun je processen beperken met betrekking tot cpu, cpu-gebruik, geheugen en vele andere zaken. Dit mechanisme wordt ook gebruikt bij containers. Voor ons probleem willen we een applicatie niet voor 100% isoleren; isoleren op geheugen zou voldoende moeten zijn.

Bij de vorige bijeenkomst richtten we onze aandacht op Wine. Inmiddels gebruikt Jan met veel plezier de Linux-applicatie *nomacs*. Afgelopen bijeenkomst vonden we Firefox verdacht. Nee, we begrijpen nog steeds niet wat Jan anders doet dan alle andere Linux-gebruikers.

Het plan is nu als volgt: maak een cgroup die aan Firefox een beperkte hoeveelheid geheugen geeft, en zorg ervoor dat je dit niet bij elke reboot handmatig opnieuw moet instellen. Firefox kan er dan niet meer voor zorgen dat de computer onbruikbaar wordt.

Als Firefox toch meer geheugen wil gebruiken dan geconfigureerd is, dan moet het operatingsysteem ingrijpen. Linux heeft hiervoor een *out-of-memory killer (oom-killer)*. De oom-killer breekt processen af als het systeem out-of-memory is. Dit gebeurt als het systeem fysiek geheugen te kort komt. De oom-killer stopt ook processen als de grenzen van een cgroup overschrijden dreigen te worden. Binnen Firefox zal dan een tabblad crashen.



1. Echte naam bekend bij de auteur en alle leden van de Linuxwerkgroep Groningen.

Linux Journal

Na enig zoeken vond ik twee websites die ik heel leerzaam vond. Op de website van het Linux Journal staat een meerdelige serie artikelen over containers en cgroups. In het eerste deel **https://www.linuxjournal.com/content/everything-you-need-know-about-linux-containers-part-i-linux-control-groups-and-process** wordt uitgelegd hoe cgroups werken, het gaat nog niet over containers.

In een voorbeeld wordt een cgroup aangemaakt die een geheugenbegrenzing heeft tot 50 Mbyte. De applicatie is een simpel bash-script dat elke minuut **'hello world'** afdrukt. Dat past in 50 Mbyte. Als dat proces in die cgroup geplaatst wordt, dan gebeurt er ogenschijnlijk niets. Dat was heel anders toen de cgroup beperkt werd tot 4 kbyte: de oom-killer greep direct in.

In de tweede helft van dat artikel staat hoe je dit beter of handiger kunt doen met de programma's uit cgroup-tools. Daar staat ook hoe je zo'n cgroup automatisch kunt laten aanmaken bij het booten via systemd.

Voor mij was dit een mooie inleiding om een en ander te leren begrijpen. Maar nu moet ik dat nog toepassen op Firefox en alle componenten die door Firefox opgestart worden. Helaas is dat wel nog iets wat ik zelf moet doen. De programma's daarvoor zijn er wel, maar Ubuntu heeft niet een simpel knopje *'beperk Firefox tot maximaal 2 Gbyte'*.

Microsoft Teams

Op de website van Remy van Elst kwam ik het volgende artikel tegen: **https://raymii.org/s/articles/Limit_specific_process_memory_on_desktop_linux_with_cgroups.html** Remy wil Microsoft Teams gebruiken onder Linux, maar hij wil er geen last van hebben als dat programma te veel geheugen gebruikt.

Earlyoom

Halverwege zijn artikel maakt Remy eerst een uitstapje naar het pakket *earlyoom*. Earlyoom is een out-of-memory killer. De manual page van earlyoom geeft aan waarom earlyoom beter is dan de standaard oom-killer van de kernel. Je kunt die manual page lezen voordat je earlyoom installeert: **http://manpages.ubuntu.com/manpages/bionic/man1/earlyoom.1.html**

Misschien dat earlyoom het verschijnsel waar Jan last van heeft al oplost.

Het artikel van Remy gaat verder over het beperken van het geheugengebruik van processen met cgroups; hoe je dat moet activeren bij het booten van de computer en hoe je MS

Teams, maar ook bijvoorbeeld Firefox, automatisch in de juiste cgroup plaatst.

Raspbian en Ubuntu

Om het geheugen te limiteren moet je, als root, het maximaal toegestane geheugengebruik schrijven naar een bestand ergens diep in */sys*. Op mijn Raspberry bleek dat bestand niet te bestaan. Dit is op te lossen door
```
cgroup_memory=1 cgroup_enable=memory
```
toe te voegen aan */boot/cmdline.txt*.

Als je bovendien het gebruik van het swap-geheugen wilt limiteren, moet je de totale geheugenlimiet *ram+swap* schrijven naar een bestand met de naam: *memory.memsw.limit_in_bytes*.
Dit bestand bestaat in Ubuntu niet. Je wilt toch echt ook het swap-geheugen limiteren, anders gaat het systeem swappen en traag worden, voordat het proces dat zich misdraagt gestopt wordt.

Op stackexchange: **https://unix.stackexchange.com/questions/44985/limit-memory-usage-for-a-single-linux-process/125024#125024** staat hoe je dit, als root, kunt oplossen:
pas */etc/default/grub* aan:
```
GRUB_CMDLINE_LINUX_DEFAULT="" →
GRUB_CMDLINE_LINUX_DEFAULT="cgroup_enable=memory
                                    swapaccount=1"
```
en voer de volgende commandos uit:
```
apt install cgroup-bin
update-grub
```
Na een reboot kun je nu ook het totale geheugengebruik ram+swap limiteren.

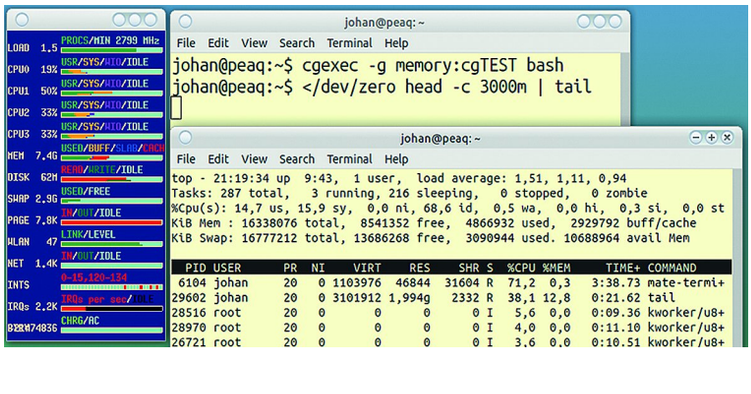
Een cgroup van 2 GByte

Ik heb een en ander uitgeprobeerd. De stappen die ik, als root, heb uitgevoerd zijn de volgende:

```
apt install cgroup-tools
cgcreate -t johan:johan -a johan:johan -g
                                    memory:/cgTEST
echo 2048m > /sys/fs/cgroup/memory/cgTEST/
                            memory.limit_in_bytes
echo 2048m > /sys/fs/cgroup/memory/cgTEST/
                        memory.memsw.limit_in_bytes
```

Dit betekent dat een cgroup gemaakt wordt, die gekoppeld is aan de Linux-gebruiker **johan**[2] en de Linux-groep **johan**. Applicaties in die cgroup mogen samen maximaal 2 GByte ram gebruiken. Ook ram+swap is gelimiteerd tot 2 GByte.

De gewone gebruiker johan kan dit testen:
```
cgexec -g memory:cgTEST bash
</dev/zero head -c 2050m | tail
```



Dit betekent dat bash wordt uitgevoerd in de cgroup cgTEST. Het programma *head* kopieert 2050 MByte nullen vanuit */dev/zero* naar de *pipe*.
Het programma *tail* zal de laatste tien regels afdrukken, maar krijgt een enkele regel van 2050 MByte aangeboden. Dus *tail* zal meer gebruiken dan de beschikbare 2 GByte . Het proces wordt inderdaad netjes door de kernel gestopt.

Als je *cgTEST* toch enige swapspace wilt geven, dan kun je *memory.memsw.limit_in_bytes* natuurlijk vergroten. Je zult dan zien dat tail bijna 2 Gbyte gaat gebruiken, en dat swapspace in gebruik genomen wordt, terwijl het geheugen van de computer nog niet vol is.

Permanent maken

Nu we dit getest hebben, kunnen we *cgTEST* permanent maken met *cgconfigparser*. Wederom als root, maak het bestand */etc/cgconfig.conf* aan met als inhoud:
```
group cgTEST {
    perm {
        admin {
            uid = johan;
        }
        task {
            uid = johan;
        }
    }
    memory {
        memory.limit_in_bytes = 2048m;
        memory.memsw.limit_in_bytes = 3072m;
    }
}
```

Let op, dit is weer gerelateerd aan de gebruiker **johan**, zoals die op mijn systeem bestaat! Kijk voor makkelijk knippen en plakken op de website van Remy. Je kunt dan ook zien wat ik heb aangepast om het voor de gebruiker johan te laten werken.

Eerst maar even *cgconfigparser* handmatig opstarten om te controleren dat er geen typfouten in de configuratiefile zitten:
```
cgconfigparser -l /etc/cgconfig.conf
```

Systemd is de moderne manier om programma's op te starten bij het booten van de computer.
Om *cgconfigparser* automatisch uit te voeren heb ik het bestand */lib/systemd/system/cgconfigparser.service* aangemaakt met de volgende inhoud:
```
[Unit]
Description=cgroup config parser
After=network.target

[Service]
User=root
Group=root
ExecStart=/usr/sbin/cgconfigparser -l /etc/
                                    cgconfig.conf
Type=oneshot

[Install]
WantedBy=multi-user.target
```

Nu nog even aan *systemd* vertellen dat deze service actief moet zijn:
```
systemctl enable cgconfigparser
systemctl start cgconfigparser
```

Nu wordt bij een reboot de cgroup *cgTEST* telkens opnieuw aangemaakt, met limieten van 2 GByte voor *ram* en 3 Gbyte voor *ram+swap*.

2. Denk eraan dat je dit aanpast voor jouw situatie.

Firefox in cgTEST

Voor mijn demonstratie moet ik er nu nog voor zorgen dat Firefox in deze cgroup geplaatst wordt. Er zijn verschillende opties:

- Firefox voortaan opstarten vanuit de commandoregel met `cgexec -g memory:cgTEST firefox`
- in een configuratiefile zoals */usr/share/applications/firefox.desktop* aangeven dat Firefox moet worden opgestart als *cgexec -g memory:cgTEST firefox*
- een *cronjob* maken die elke minuut alle Firefox-processen in de juiste cgroup stopt met `cgclassify -g memory:cgTEST $(pidof firefox)`
- er is een *'control group rules engine daemon': cgrulesengd*. Dit is iets fraaier, en waarschijnlijk ook efficiënter en effectiever dan de *cronjob*.

Het artikel van Remy eindigt met de beschrijving van *cgrulesengd*. Daar heb ik voor gekozen: ik heb eerst als root de configuratie-file */etc/cgrules.conf voor cgrulesend* gemaakt. Deze file moet uit 1 regel bestaan
`johan:firefox memory cgTEST/`

Deze regel zorgt ervoor dat "firefox" van de gebruiker **johan** in de cgroup *cgTEST* komt. De stappen om ervoor te zorgen dat *cgrulesengd* opstart bij het booten van de computer zijn, als root: maak het bestand */lib/systemd/system/cgrulesgend.service* aan met als inhoud:

```
[Unit]
Description=cgroup rules generator
After=network.target cgconfigparser.service

[Service]
User=root
Group=root
Type=forking
EnvironmentFile=-/etc/cgred.conf
ExecStart=/usr/sbin/cgrulesengd
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Kopieer een standaard configuratiefile, en vertel systemd dat cgrulesengd als daemon moet blijven draaien:

```
cp /usr/share/doc/cgroup-tools/examples/cg-
red.conf /etc/cgred.conf
systemctl enable cgrulesgend
systemctl start cgrulesgend
```

En inderdaad, na een reboot werden alle Firefox-processen in *cgTEST* geplaatst.

Nog een paar losse opmerkingen over deze cgroup.
In */sys/fs/cgroup/memory/cgTEST/tasks* maar ook in */sys/fs/cgroup/memory/cgTEST/cgroup.procs* staan de proces-ids van alle processen in deze cgroup.

Als een proces al draait, dan kan root het achteraf alsnog in een cgroup stoppen:

```
echo $PID > /sys/fs/cgroup/memory/cgTEST/
                                    cgroup.procs
cgclassify -g memory:cgTEST lijst_van_proces_ids
```