

*Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών
Υπολογιστών*



8ο εξάμηνο

2017-2018

Εργαστήριο Λειτουργικών Συστημάτων

1^η Εργαστηριακή Άσκηση

Επιτήρηση χρήσης πόρων εφαρμογών με Linux Cgroups

Μουζάκης Ανάργυρος-Γεώργιος

Μουζάκης Νικόλαος

Εισαγωγή

Η άσκηση αφορά στην διαχείριση των πόρων του cpu subsystem με χρήση του συστήματος *linux cgroups*. Συγκεκριμένα, μας δόθηκε ο δαίμονας *cgmond*, σκοπός του οποίου είναι η διαχείριση των πόρων που προσφέρει ένα data center σε πελάτες. Προκειμένου να επιτευχθεί αυτή η λειτουργία, ο δαίμονας αλληλεπιδρά με τις ιεραρχίες *cpu* και *monitor*. Συγκεκριμένα, χρησιμοποιώντας την ιεραρχία *cpu* μπορούν να γίνουν κατάλληλες ρυθμίσεις προκειμένου, για τις εφαρμογές που εκτελούνται, να εξασφαλιστεί ότι θα έχουν τουλάχιστον ένα μέρος της ισχύος του επεξεργαστή. Επιπλέον, μέσω της ιεραρχίας *monitor* ρυθμίζονται θέματα σχετικά με την εισαγωγή εφαρμογών σε *cgroups*, όταν αυτές ξεκινούν να εκτελούνται, καθώς και με τη διαγραφή τους, όταν ολοκληρώνουν την εκτέλεσή τους.

Ο δαίμονας δεν αλληλεπιδρά απευθείας με τις δύο παραπάνω ιεραρχίες, αλλά μέσω δύο python scripts τα οποία γράψαμε εμείς. Πρόκειται για τα *mod_policy_cpumin.py* και *mod_limit_cpu.py*. Λεπτομερής περιγραφή των λειτουργιών που αυτά επιτελούν ακολουθεί στις επόμενες σελίδες.

mod policy cpumin.py

Ο κώδικας αυτού του script είναι ο ακόλουθος:

```
1. #!/usr/bin/env python
2. import sys
3.
4. total=2000 #ta synolika xiliosta
5.
6. lines=sys.stdin.readlines()
7. apps=[]
8. reqs=[]
9. for line in lines:
10.     args=line.split(":")
11.     apps.append(args[1])
12.     reqs.append(int(args[3].rstrip()))
13.
14. if sum(reqs)>0:
15.     mult=total/(1.0*sum(reqs))
16. else:
17.     mult=1
18. if mult<1:
19.     score=-1
20. else:
21.     score=1
22.
23. sys.stdout.write("score:"+str(score)+"\n")
24.
25. reqs=[i*mult for i in reqs]
26.
27. for req,app in zip(reqs,apps):
28.     shares=int((req/(1.0*total))*10000)
29.     sys.stdout.write("set_limit:" + app + ":cpu.shares:" + str((shares))+ "\n")
```

Το εν λόγω script αλληλεπιδρά με την ιεραρχία cpu και εκτελείται κάθε φορά που γίνεται spawn μία καινούρια εφαρμογή. Στόχος της λειτουργίας του είναι να ελέγχει αν είναι δυνατή η εισαγωγή της καινούριας εφαρμογής. Για αυτό το λόγο, υπολογίζει τα χιλιοστά που απαιτούν συνολικά οι εφαρμογές που ήδη εκτελούνται και προσθέτει σε αυτά εκείνα της καινούριας. Εφόσον το άθροισμα που θα προκύψει είναι μικρότερο από 2000 (το σύστημα όπου υλοποιούνται οι ασκήσεις έχει δύο επεξεργαστές, με τον καθένα να προσφέρει 1000 χιλιοστά), η καινούρια εφαρμογή μπαίνει και αυτή στο σύστημα για εκτέλεση. Τέλος, τα απαιτούμενα χιλιοστά των υπό εκτέλεση εφαρμογών μετατρέπονται σε cpu.shares. Ο κώδικας είναι δομημένος έτσι

ώστε η είσοδος και η έξοδος να συμφωνούν με τις προδιαγραφές που αναφέρονται στην εκφώνηση της άσκησης.

Εδώ πρέπει να αναφερθούν δύο πράγματα. Το πρώτο έχει να κάνει με τη συμπεριφορά εφαρμογών που ζητούν παραπάνω χιλιοστά από αυτά που προσφέρει ένας μεμονωμένος επεξεργαστής. Συγκεκριμένα, αν μία εφαρμογή απαιτεί 2000 χιλιοστά επεξεργαστικής ισχύος, θα δρομολογηθεί, εφόσον δεν εκτελείται κάποια άλλη. Όμως, εφόσον ο κώδικας της εφαρμογής δεν είναι παραλληλοποιήσιμος (πχ να γίνεται χρήση πολλαπλών threads κλπ), στην πράξη, η εφαρμογή θα αξιοποιεί μόνο 1000 χιλιοστά, αφού δεν θα μπορεί να εκτελείται παράλληλα σε δύο επεξεργαστές. Παρότι, όμως, δεν αξιοποιούνται πλήρως οι δυνατότητες του συστήματος, δεν θα μπορούν να εκτελεστούν άλλες εφαρμογές, αφού τα συνολικά απαιτούμενα χιλιοστά θα προκύπτουν περισσότερα από 2000. Μία τέτοια περίπτωση φαίνεται παρακάτω.

```
Server starting... OK

sleep 1

cgmon app list
-----
| App |
| No apps |
-----

sleep 1

# assuming total 2000 millicpus == 2 cpus
cgmon policy create -n platinum -p 2000
cgmon policy create -n elastic -p 250
cgmon policy list
-----
| Name | cpu |
|-----|-----|
| default_min100 | 100 |
| default_min1000 | 1000 |
| default_min500 | 500 |
| elastic | 250 |
| platinum | 2000 |
-----

sleep 1

cgmon app spawn -p platinum -e "/root/cgmon/anel.sh" -n BANKDB
cgmon app spawn -p elastic -e "stress -c 2" -n VIDEOENC
ERROR::Server returned: 'Resources 'CPU' returned negative scores'
cgmon app spawn -p elastic -e "stress -c 2" -n SPAMBOT
ERROR::Server returned: 'Resources 'CPU' returned negative scores'
cgmon app list
-----
| App |
| BANKDB |
|-----|

sleep 5

sleep 1000000
]
```

```
1 100.0% Tasks: 30, 9 thr: 2 running
2 0.0% Load average: 0.40 0.40 0.40
Max 110/399500U Uptime: 4 days, 22:44:03
Sum 0/0000

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
6478 root 20 0 7168 92 0 R 99.8 0.0 0:04.67 stress -c 1 -t 10 -d
6346 root 20 0 2476 3624 948 R 0.9 0.1 0:01.16 htop
6443 root 20 0 1036 10936 4756 S 0.5 0.4 0:00.59 /usr/bin/python /usr
6331 root 20 0 2912 1080 516 S 0.0 0.1 0:00.71 tmux
6445 root 20 0 1254 15936 4756 S 0.0 0.4 0:00.05 /usr/bin/python /usr
4948 root 20 0 6292 6052 128 S 0.0 0.1 0:01.55 sshd: root@pts/0
6463 root 20 0 1244 1024 792 S 0.0 0.1 0:00.50 /bin/bash /root/cgmo
6332 root 20 0 2244 5204 316 S 0.0 0.1 0:00.19 -bash
1 root 20 0 2656 4528 944 S 0.0 0.1 0:14.95 /sbin/init
369 root 20 0 3384 4232 624 S 0.0 0.1 1:02.55 /usr/sbin/ntpd -p /v
360 root 20 0 4272 3404 900 S 0.0 0.1 0:05.83 /usr/bin/dbus-daemon
140 root 20 0 3964 6908 624 S 0.0 0.2 0:24.60 /lib/systemd/systemd
352 root 20 0 400 12852 1720 S 0.0 0.3 0:10.79 /usr/sbin/NetworkMan
397 root 20 0 2400 10036 1132 S 0.0 0.2 0:00.57 /sbin/dhclient -d -q
152 root 20 0 4008 3340 616 S 0.0 0.1 0:00.35 /lib/systemd/systemd
383 root 20 0 4208 12852 10720 S 0.0 0.3 0:00.00 /usr/sbin/NetworkMan
387 root 20 0 4208 12852 10720 S 0.0 0.3 0:00.00 /usr/sbin/NetworkMan
388 root 20 0 4208 12852 10720 S 0.0 0.3 0:00.04 /usr/sbin/NetworkMan
353 root 20 0 2476 2768 528 S 0.0 0.1 0:01.19 /usr/sbin/cron -f
355 root 20 0 2656 2764 432 S 0.0 0.1 0:02.09 /lib/systemd/systemd
380 root 20 0 2528 3416 604 S 0.0 0.1 0:02.08 /usr/sbin/rsyslogd -
381 root 20 0 2528 3416 604 S 0.0 0.1 0:00.00 /usr/sbin/rsyslogd -
382 root 20 0 2528 3416 604 S 0.0 0.1 0:02.53 /usr/sbin/rsyslogd -
370 root 20 0 2528 3416 604 S 0.0 0.1 0:04.71 /usr/sbin/rsyslogd -
371 root 20 0 10276 1996 764 S 0.0 0.0 0:57.60 /usr/sbin/irqbalance
374 root 20 0 4256 1572 1420 S 0.0 0.0 0:00.01 /usr/sbin/acpid
377 root 20 0 58184 5160 4488 S 0.0 0.1 0:06.03 /usr/sbin/sshd -D
379 root 20 0 14416 1944 1792 S 0.0 0.0 0:00.00 /sbin/agetty --nocle
391 root 20 0 2710 5580 4912 S 0.0 0.1 0:00.19 /usr/lib/policykit-1
392 root 20 0 2710 5580 4912 S 0.0 0.1 0:00.00 /usr/lib/policykit-1
389 root 20 0 2710 5580 4912 S 0.0 0.1 0:00.35 /usr/lib/policykit-1
4953 root 20 0 2204 5096 3252 S 0.0 0.1 0:00.13 -bash
6329 root 20 0 1260 2812 508 S 0.0 0.1 0:00.00 tmux
6330 root 20 0 2232 4936 1124 S 0.0 0.1 0:00.08 -bash
28526 root 20 0 2088 3340 896 S 0.0 0.1 0:00.02 /lib/systemd/systemd
28527 root 20 0 4936 1680 0 S 0.0 0.0 0:00.00 (sd-pam)
6437 root 20 0 1244 1024 792 S 0.0 0.1 0:00.00 /bin/bash ./mydemo.s
F1:top F2:help F3:search F4:filter F5:tree F6:sort F7:kill F8:info F9:kill F10:exit
```

Συγκεκριμένα, πρώτα γίνεται spawn μία εφαρμογή με ελάχιστα απαιτούμενα χιλιοστά 2000. Ο κώδικας αυτής της εφαρμογής είναι:

1. `#!/bin/bash`
- 2.
3. `while true`
4. `do`
5. `stress -c 1 -t 10 -q`
6. `sleep 10`
7. `done`

Ανά 10 δευτερόλεπτα, ο ένας εκ των δύο επεξεργαστών επιβαρύνεται από την εφαρμογή. Μετά ακολουθεί μία περίοδος 10 δευτερολέπτων, στη διάρκεια των οποίων η εφαρμογή παραμένει αδρανής. Αυτή η συμπεριφορά, σε συνδυασμό με τις απαιτήσεις σε χιλιοστά, προσδίδουν στην εν λόγω εφαρμογή όλα τα στοιχεία μίας ανελαστικής εφαρμογής.

Τα apps τα οποία γίνονται spawn πιο μετά απορρίπτονται από τον επεξεργαστή, λόγω του περιορισμού που υπάρχει για τα συνολικά χιλιοστά.

Το δεύτερο πράγμα που πρέπει να σχολιαστεί αφορά στη διαχείριση των πόρων σε περίπτωση που τα απαιτούμενα χιλιοστά είναι λιγότερα από τα διαθέσιμα. Συγκεκριμένα, μία πολιτική που εφαρμόζεται συχνά σε τέτοιες περιπτώσεις είναι τα επιπλέον χιλιοστά να παραχωρούνται σε ελαστικές εφαρμογές (δηλαδή, εφαρμογές με μικρές απαιτήσεις σε χιλιοστά που δεν έχουν τις διακυμάνσεις της εφαρμογής που παρουσιάστηκε παραπάνω). Αντί για αυτό, εμείς προτιμήσαμε τα πλεονασματικά χιλιοστά να κατανέμονται αναλογικά σε όλες τις εφαρμογές.

Δύο τέτοιες περιπτώσεις φαίνονται παρακάτω, όπου έχουμε λιγότερα συνολικά χιλιοστά από τα απαιτούμενα.

```
Server starting... OK
sleep 1
cgmon app list
+-----+
| App |
+-----+
| No apps |
+-----+

sleep 1
# assuming total 2000 millicpus == 2 cpus
cgmon policy create -n platinum -p 500
cgmon policy create -n elastic -p 250
cgmon policy list
+-----+
| Name | cpu |
+-----+
| default_min100 | 100 |
| default_min1000 | 1000 |
| default_min500 | 500 |
| elastic | 250 |
| platinum | 500 |
+-----+

sleep 1
cgmon app spawn -p platinum -e "/root/cgmon/anel.sh" -n BANKDB
cgmon app spawn -p elastic -e "stress -c 2" -n VIDEOENC
cgmon app spawn -p elastic -e "stress -c 2" -n SPAMBOT
cgmon app list
+-----+
| App |
+-----+
| SPAMBOT |
| BANKDB |
| VIDEOENC |
+-----+

sleep 5
sleep 1000000
[1] B.bash*
```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
6845	root	20	0	7168	88	0	R	99.5	0.0	0:03.30	stress -c 1 -t 10 -g
6838	root	20	0	172	92	0	R	25.1	0.0	0:15.50	stress -c 2
6826	root	20	0	172	92	0	R	25.1	0.0	0:15.61	stress -c 2
6837	root	20	0	172	92	0	R	24.6	0.0	0:15.51	stress -c 2
6827	root	20	0	172	92	0	R	24.6	0.0	0:15.68	stress -c 2
371	root	20	0	10276	1996	764	S	0.5	0.0	0:57.74	/usr/sbin/irqbalance
6608	root	20	0	24476	636	960	S	0.0	0.1	0:01.05	htop
6595	root	20	0	24928	484	520	S	0.0	0.1	0:00.55	tmux
6780	root	20	0	12500	15028	752	S	0.0	0.4	0:00.61	/usr/bin/python /usr
4948	root	20	0	9292	6052	128	S	0.0	0.1	0:01.88	sshd: root@pts/0
6782	root	20	0	12500	15028	752	S	0.0	0.4	0:00.07	/usr/bin/python /usr
6799	root	20	0	12444	904	672	S	0.0	0.1	0:00.44	/bin/bash /root/cgmo
6815	root	20	0	172	940	856	S	0.0	0.0	0:00.25	stress -c 2
6831	root	20	0	172	952	868	S	0.0	0.0	0:00.25	stress -c 2
6774	root	20	0	12444	2968	736	S	0.0	0.1	0:00.01	/bin/bash ./mydemo.s
6596	root	20	0	2236	5196	316	S	0.0	0.1	0:00.13	-bash
380	root	20	0	2520	3416	604	S	0.0	0.1	0:02.10	/usr/sbin/rsyslogd -
1	root	20	0	2656	4520	944	S	0.0	0.1	0:15.03	/sbin/init
140	root	20	0	3964	4076	602	S	0.0	0.2	0:24.69	/lib/systemd/systemd
352	root	20	0	4200	12852	1720	S	0.0	0.3	0:10.85	/usr/sbin/NetworkMan
369	root	20	0	3384	4232	624	S	0.0	0.1	1:02.96	/usr/sbin/ntpd -p /v
152	root	20	0	4008	3340	616	S	0.0	0.1	0:00.35	/lib/systemd/systemd
303	root	20	0	4200	12852	1720	S	0.0	0.3	0:00.00	/usr/sbin/NetworkMan
387	root	20	0	4200	12852	1720	S	0.0	0.3	0:00.00	/usr/sbin/NetworkMan
388	root	20	0	4200	12852	1720	S	0.0	0.3	0:00.04	/usr/sbin/NetworkMan
353	root	20	0	2476	2768	528	S	0.0	0.1	0:01.20	/usr/sbin/cron -f
355	root	20	0	2356	2764	432	S	0.0	0.1	0:02.92	/lib/systemd/systemd
360	root	20	0	4222	404	900	S	0.0	0.1	0:05.06	/usr/bin/dbus-daemon
381	root	20	0	2520	3416	604	S	0.0	0.1	0:00.00	/usr/sbin/rsyslogd -
382	root	20	0	2520	3416	604	S	0.0	0.1	0:02.54	/usr/sbin/rsyslogd -
370	root	20	0	2520	3416	604	S	0.0	0.1	0:04.71	/usr/sbin/rsyslogd -
374	root	20	0	256	1572	420	S	0.0	0.0	0:00.01	/usr/sbin/acpid
377	root	20	0	1416	160	488	S	0.0	0.1	0:06.04	/usr/sbin/sshd -D
379	root	20	0	1416	1944	792	S	0.0	0.0	0:00.00	/sbin/agetty --nicle
391	root	20	0	2710	5580	912	S	0.0	0.1	0:00.19	/usr/lib/policykit-1
392	root	20	0	2710	5580	912	S	0.0	0.1	0:00.00	/usr/lib/policykit-1
389	root	20	0	2710	5580	912	S	0.0	0.1	0:00.36	/usr/lib/policykit-1

F1:help F2:setup F3:search F4:filter F5:rec F6:sortby F7:live F8:Nice F9:kill F10:quit

Σε αυτή την περίπτωση η ανελαστική εφαρμογή απαιτεί 500 χιλιοστά, ενώ υπάρχουν 2 ελαστικές, καθεμία εκ των οποίων απαιτεί 250. Το αποτέλεσμα είναι πως στην μεν πρώτη δίνεται το 100% ενός επεξεργαστή, οι δε άλλες μοιράζονται τον άλλο, με την καθεμία να παίρνει από 50% (στο htop βλέπουμε 4 των 25%, επειδή κάθε εφαρμογή κάνει spawn δύο workers που απασχολούν τη cpu).

```
Server starting... OK
sleep 1
cgmon app list
+-----+
| App |
+-----+
| No apps |
+-----+

sleep 1
# assuming total 2000 millicpus == 2 cpus
cgmon policy create -n platinum -p 500
cgmon policy create -n elastic -p 250
cgmon policy list
+-----+
| Name | cpu |
+-----+
| default_min100 | 100 |
| default_min1000 | 1000 |
| default_min5000 | 500 |
| elastic | 250 |
| platinum | 500 |
+-----+

sleep 1
cgmon app spawn -p platinum -e "/root/cgmon/anel.sh" -n BANKDB
cgmon app spawn -p elastic -e "stress -c 2" -n VIDEOENC
cgmon app spawn -p elastic -e "stress -c 2" -n SPAMBOT
cgmon app list
+-----+
| App |
+-----+
| SPAMBOT |
| BANKDB |
| VIDEOENC |
+-----+

sleep 5
sleep 1000000
[1] 8888888
```

1	[]	100.0%	Tasks: 35, 9 thr, 5 running
2	[]	100.0%	Load average: 1.37
Mem	[]	117/3905MB	Uptime: 4 days, 23:33:58
Swp	[]	0/0MB	

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
6826	root	20	0	7172	92	0	R	49.9	0.0	0:11.21	stress -c 2
6838	root	20	0	172	92	0	R	49.9	0.0	0:11.08	stress -c 2
6837	root	20	0	172	92	0	R	49.9	0.0	0:11.07	stress -c 2
6827	root	20	0	172	92	0	R	49.9	0.0	0:11.24	stress -c 2
6595	root	20	0	21928	3084	520	S	0.5	0.1	0:00.55	tmux
6780	root	20	0	125M	15028	4752	S	0.0	0.4	0:00.61	/usr/bin/python /usr
4948	root	20	0	9292	6052	128	S	0.0	0.1	0:01.88	sshd: root@pts/0
6608	root	20	0	2476	856	968	S	0.0	0.1	0:01.02	htop
6782	root	20	0	1244	15028	752	S	0.0	0.4	0:00.07	/usr/bin/python /usr
6799	root	20	0	1244	2004	672	S	0.0	0.1	0:00.44	/bin/bash /root/cgmo
6815	root	20	0	172	940	856	S	0.0	0.0	0:00.25	stress -c 2
6831	root	20	0	172	952	868	S	0.0	0.0	0:00.25	stress -c 2
6774	root	20	0	1244	2068	736	S	0.0	0.1	0:00.01	/bin/bash ./mydemo.s
6596	root	20	0	2236	5196	316	S	0.0	0.1	0:00.13	-bash
380	root	20	0	252M	3416	604	S	0.0	0.1	0:02.10	/usr/sbin/rsyslogd -
1	root	20	0	2656	4528	944	S	0.0	0.1	0:15.03	/sbin/init
140	root	20	0	31964	6976	602	S	0.0	0.2	0:24.69	/lib/systemd/systemd
352	root	20	0	420M	12852	1720	S	0.0	0.3	0:10.85	/usr/sbin/NetworkMan
369	root	20	0	3384	4232	624	S	0.0	0.1	1:02.06	/usr/sbin/ntpd -o /v
152	root	20	0	4108	3340	616	S	0.0	0.1	0:00.35	/lib/systemd/systemd
383	root	20	0	420M	12852	1720	S	0.0	0.3	0:00.00	/usr/sbin/NetworkMan
387	root	20	0	420M	12852	1720	S	0.0	0.3	0:00.00	/usr/sbin/NetworkMan
388	root	20	0	420M	12852	1720	S	0.0	0.3	0:00.04	/usr/sbin/NetworkMan
353	root	20	0	2476	2760	528	S	0.0	0.1	0:01.20	/usr/sbin/cron -f
355	root	20	0	28356	2764	432	S	0.0	0.1	0:02.02	/lib/systemd/systemd
360	root	20	0	4272	3404	900	S	0.0	0.1	0:05.86	/usr/bin/dbus-daemon
381	root	20	0	252M	3416	604	S	0.0	0.1	0:00.00	/usr/sbin/rsyslogd -
382	root	20	0	252M	3416	604	S	0.0	0.1	0:02.54	/usr/sbin/rsyslogd -
370	root	20	0	252M	3416	604	S	0.0	0.1	0:04.71	/usr/sbin/rsyslogd -
371	root	20	0	10276	1096	764	S	0.0	0.0	0:57.73	/usr/sbin/irqbalance
374	root	20	0	256	1572	420	S	0.0	0.0	0:00.01	/usr/sbin/acpid
377	root	20	0	5184	5160	488	S	0.0	0.1	0:06.04	/usr/sbin/sshd -D
379	root	20	0	1416	1044	792	S	0.0	0.0	0:00.00	/sbin/agetty --nocle
391	root	20	0	271M	5580	912	S	0.0	0.1	0:00.19	/usr/lib/policykit-1
392	root	20	0	271M	5580	912	S	0.0	0.1	0:00.00	/usr/lib/policykit-1
389	root	20	0	271M	5580	912	S	0.0	0.1	0:00.36	/usr/lib/policykit-1
397	root	20	0	2400	10036	132	S	0.0	0.2	0:00.57	/sbin/dhclient -d -q

Εδώ η ανελαστική εφαρμογή είναι σε περίοδο αδράνειας, οπότε οι 2 ελαστικές έχουν η καθεμία από 1 επεξεργαστή που απασχολούν κατά 100%.

mod limit cpu.py

Ο κώδικας αυτού του script είναι ο ακόλουθος:

```
1. #!/usr/bin/env python
2. import sys
3. import os
4.
5. lines = sys.stdin.readlines()
6.
7. file=open("/root/cgmon/LOG.txt","w")
8.
9. for line in lines:
10.     args=line.split(":")
11.     if args[0]=='create':
12.         os.system('mkdir -p /sys/fs/cgroup/cpu/monitor/' +args[3].rstrip('\n'))
13.         file.write('mkdir -p /sys/fs/cgroup/cpu/monitor/' +args[3].rstrip('\n')+'\n')
14.     elif args[0]=='remove':
15.         os.system('rmdir /sys/fs/cgroup/cpu/monitor/' +args[3].rstrip('\n'))
16.         file.write('rmdir /sys/fs/cgroup/cpu/monitor/' +args[3].rstrip('\n')+'\n')
17.     elif args[0]=='add':
18.         os.system('echo '+args[4].rstrip('\n')+' >
/sys/fs/cgroup/cpu/monitor/' +args[3]+'/tasks')
19.         file.write('echo '+args[4].rstrip('\n')+' >
/sys/fs/cgroup/cpu/monitor/' +args[3]+'/tasks'+'\n')
20.     elif args[0]=='set_limit':
21.         os.system('echo '+args[5].rstrip('\n')+' >
/sys/fs/cgroup/cpu/monitor/' +args[3]+'/cpu.shares')
22.         file.write('echo '+args[5].rstrip('\n')+' >
/sys/fs/cgroup/cpu/monitor/' +args[3]+'/cpu.shares'+'\n')
23.     else:
24.         file.write('empty!\n')
25. file.close()
```

Αυτό το script αλληλεπιδρά με την ιεραρχία monitor, προκειμένου να εφαρμόσει τις ρυθμίσεις που παρήγαγε το προηγούμενο script. Παραπάνω φαίνονται οι εντολές που εκτελούνται σε καθεμία από τις πέντε περιπτώσεις που αναφέρονται στην εκφώνηση. Η πρώτη περίπτωση αντιστοιχεί στη δημιουργία ενός νέου cgroup. Η δεύτερη στη διαγραφή μίας εφαρμογής από ένα cgroup. Η τρίτη στην προσθήκη μίας εφαρμογής σε ένα υπάρχον cgroup. Η τέταρτη στην ρύθμιση των cpu.shares μίας εφαρμογής. Όταν γίνεται spawn μία εφαρμογή, εκτελούνται οι ενέργειες που αντιστοιχούν στην τρίτη και στην τέταρτη περίπτωση και, ενδεχομένως, στην πρώτη. Όταν ολοκληρώνεται μία εφαρμογή έχουμε την δεύτερη περίπτωση.

Κάθε εντολή που εκτελείται γράφεται και σε ένα αρχείο *LOG.txt*, γεγονός που μας βοήθησε στο debugging κατά τη συγγραφή του script. Στο debugging βοήθησε και η ύπαρξη της εντολής *else: file.write('empty!\n')*.

Επιπλέον παραδείγματα

Παρακάτω φαίνονται τα αποτελέσματα μερικών ακόμη δοκιμών. Η πρώτη γίνεται με το δικό μας demo, που χρησιμοποιήθηκε και στα προηγούμενα, με τη διαφορά ότι τώρα δίνουμε 1000 χιλιοστά στην ανελαστική εφαρμογή. Το δεύτερο βασίζεται στο demo που δόθηκε ως βοηθητικό αρχείο.

```
Server starting... OK

sleep 1

cgmon app list
+-----+
| App |
+-----+
| No apps |
+-----+

sleep 1

# assuming total 2000 millicpus == 2 cpus
cgmon policy create -n platinum -p 1000
cgmon policy create -n elastic -p 250
cgmon policy list
+-----+
| Name | cpu |
+-----+
| default_min100 | 100 |
| default_min1000 | 1000 |
| default_min500 | 500 |
| elastic | 250 |
| platinum | 1000 |
+-----+

sleep 1

cgmon app spawn -p platinum -e "/root/cgmon/anel.sh" -n BANKDB
cgmon app spawn -p elastic -e "stress -c 2" -n VIDEOENC
cgmon app spawn -p elastic -e "stress -c 2" -n SPAMBOT
cgmon app list
+-----+
| App |
+-----+
| SPAMBOT |
| BANKDB |
| VIDEOENC |
+-----+

sleep 5

sleep 1000000

[1] 0.000000
```

```
1 [|||||] 100.00% Tasks: 36 9 thr: 7 running
2 [|||||] 100.00% Load average: 0.28 0.28 0.37
Mem [|||||] 117/3965MB Uptime: 5 days, 00:03:45
Swap [|||||] 0/0MB

PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
7127 root 20 0 7168 92 0 R 99.8 0.0 0:08.12 stress -c 1 -t 10 -d
7156 root 20 0 172 92 0 R 25.2 0.0 0:01.85 stress -c 2
7158 root 20 0 172 96 0 R 25.2 0.0 0:01.80 stress -c 2
7159 root 20 0 172 96 0 R 25.2 0.0 0:01.80 stress -c 2
7155 root 20 0 172 92 0 R 24.7 0.0 0:01.84 stress -c 2
6608 root 20 0 24476 3636 960 R 0.5 0.1 0:08.64 httpd
4948 root 20 0 91292 6052 128 S 0.0 0.1 0:02.25 sshd: root@pts/0
7101 root 20 0 125M 15908 4736 S 0.0 0.4 0:00.33 /usr/bin/python /usr
6595 root 20 0 21928 3168 520 S 0.0 0.1 0:03.87 tmux
7135 root 20 0 172 952 868 S 0.0 0.0 0:00.30 stress -c 2
7149 root 20 0 172 912 824 S 0.0 0.0 0:00.28 stress -c 2
7103 root 20 0 125M 15908 4736 S 0.0 0.4 0:00.06 /usr/bin/python /usr
7120 root 20 0 1244 2400 228 S 0.0 0.1 0:00.43 /bin/bash /root/cgmo
6596 root 20 0 21240 5200 316 S 0.0 0.1 0:00.19 -bash
1 root 20 0 21656 4520 944 S 0.0 0.1 0:15.07 /sbin/init
369 root 20 0 3384 4232 624 S 0.0 0.1 1:03.24 /usr/sbin/ntpd -p /v
360 root 20 0 4272 3404 900 S 0.0 0.1 0:05.88 /usr/bin/dbus-daemon
140 root 20 0 3964 7004 720 S 0.0 0.2 0:24.76 /lib/systemd/systemd
370 root 20 0 2520 3416 604 S 0.0 0.1 0:04.74 /usr/sbin/rsyslogd -
382 root 20 0 2520 3416 604 S 0.0 0.1 0:02.55 /usr/sbin/rsyslogd -
355 root 20 0 20356 2764 432 S 0.0 0.1 0:02.93 /lib/systemd/systemd
352 root 20 0 420M 12852 1720 S 0.0 0.3 0:10.89 /usr/sbin/NetworkMan
371 root 20 0 10276 1996 1764 S 0.0 0.0 0:58.09 /usr/sbin/irqbalance
377 root 20 0 53184 5160 488 S 0.0 0.1 0:06.07 /usr/sbin/sshd -D
380 root 20 0 2520 3416 604 S 0.0 0.1 0:02.10 /usr/sbin/rsyslogd -
152 root 20 0 41008 3340 616 S 0.0 0.1 0:00.35 /lib/systemd/systemd
383 root 20 0 420M 12852 1720 S 0.0 0.3 0:00.00 /usr/sbin/NetworkMan
387 root 20 0 420M 12852 1720 S 0.0 0.3 0:00.00 /usr/sbin/NetworkMan
388 root 20 0 420M 12852 1720 S 0.0 0.3 0:00.04 /usr/sbin/NetworkMan
353 root 20 0 21476 2768 528 S 0.0 0.1 0:01.20 /usr/sbin/cron -f
381 root 20 0 2520 3416 604 S 0.0 0.1 0:00.00 /usr/sbin/rsyslogd -
374 root 20 0 2556 1572 420 S 0.0 0.0 0:00.01 /usr/sbin/acpid
379 root 20 0 14416 1944 1792 S 0.0 0.0 0:00.00 /sbin/agetty --nocle
391 root 20 0 271M 5580 912 S 0.0 0.1 0:00.19 /usr/lib/policykit-1
392 root 20 0 271M 5580 912 S 0.0 0.1 0:00.00 /usr/lib/policykit-1
389 root 20 0 271M 5580 912 S 0.0 0.1 0:00.36 /usr/lib/policykit-1
397 root 20 0 2400 10036 132 S 0.0 0.2 0:00.57 /sbin/dhclient -d -q
F1Help F2Setup F3Search F4Filter F5Tree F6SortBy F7Nice F8Nice F9Kill F10Quit
```

Εδώ έχουμε άθροισμα χιλιοστών ίσο με το διαθέσιμο, οπότε όλες οι εφαρμογές παίρνουν ακριβώς τα χιλιοστά που ζητάνε.

```
# assuming total 2000 millicpus == 2 cpus
cgmon policy create -n platinum -p 1000
cgmon policy create -n silver -p 500
cgmon policy create -n elastic -p 50
cgmon policy list
+-----+
| Name | | cpu |
+-----+
| default_min100 | 100 |
| default_min1000 | 1000 |
| default_min500 | 500 |
| elastic | 50 |
| platinum | 1000 |
| silver | 500 |
+-----+
cgmon app spawn -p platinum -e "stress -c 2" -n BANKDB
cgmon app spawn -p silver -e "stress -c 2" -n WEBDB
cgmon app spawn -p elastic -e "stress -c 2" -n VIDEOENC
cgmon app spawn -p elastic -e "stress -c 2" -n SPAMBOT
cgmon app list
+-----+
| App |
+-----+
| WEBDB |
| BANKDB |
| VIDEOENC |
| SPAMBOT |
+-----+
sleep 1
cgmon app spawn -p platinum -e "stress -c 2" -n MEDICALDB
ERROR::Server returned: 'Resources 'CPU' returned negative scores'
sleep 1
# forcing it violates policies
cgmon app spawn -p platinum -e "stress -c 2" -n MEDICALDB -f
sleep 10
```

Εδώ φαίνεται πώς, αρχικά, δεν επιτρέπεται σε μία εφαρμογή να εκτελεστεί, αλλά μετά, χρησιμοποιώντας το flag f, η πολιτική αυτή παραβιάζεται και, αναγκαστικά, αναπροσαρμόζονται οι τιμές των cpu.shares κάθε εφαρμογής.