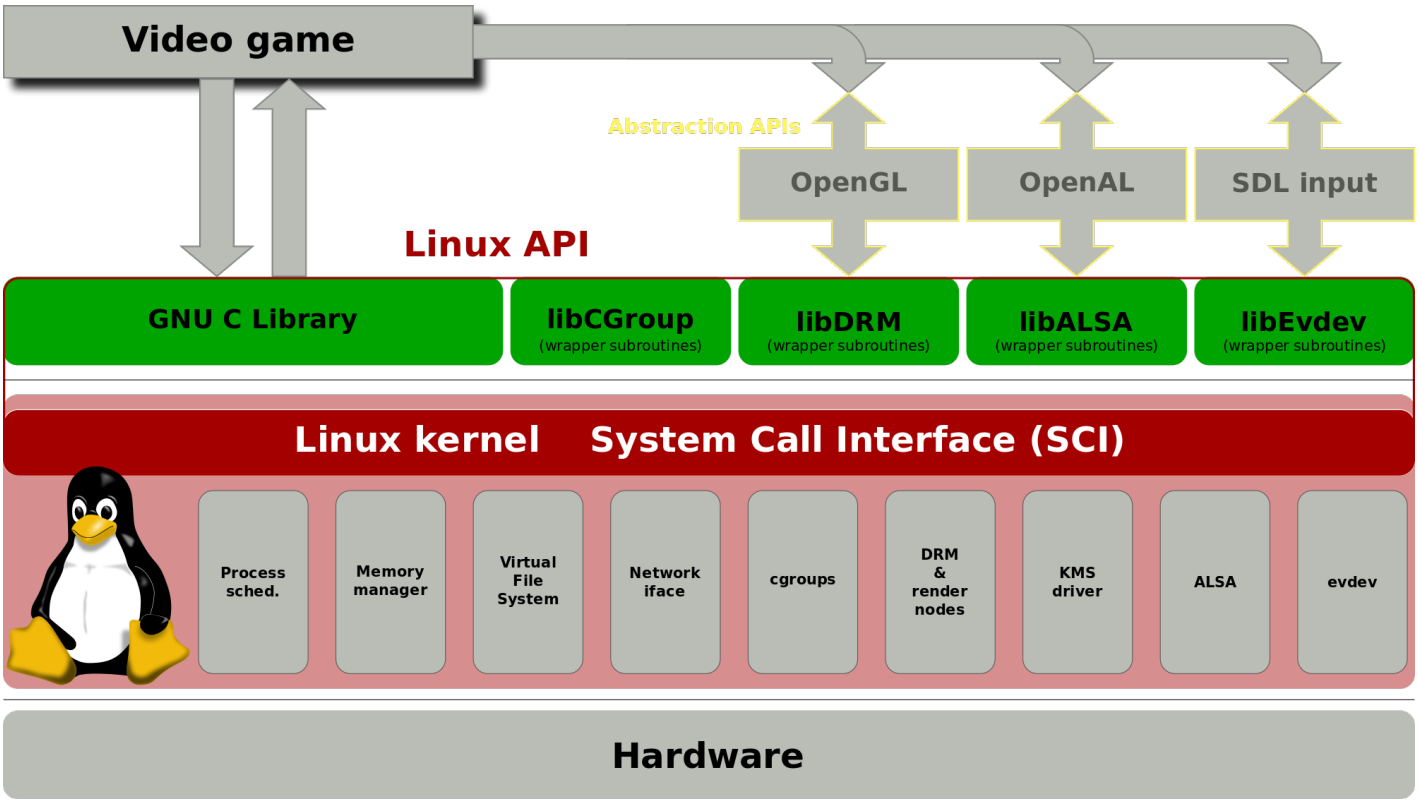
Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

1η Εργαστηριακή Άσκηση : Επιτήρηση χρήσης πόρων εφαρμογών με Linux Cgroups



Εργαστήριο Λειτουργικών Συστημάτων

Ροη : Υ

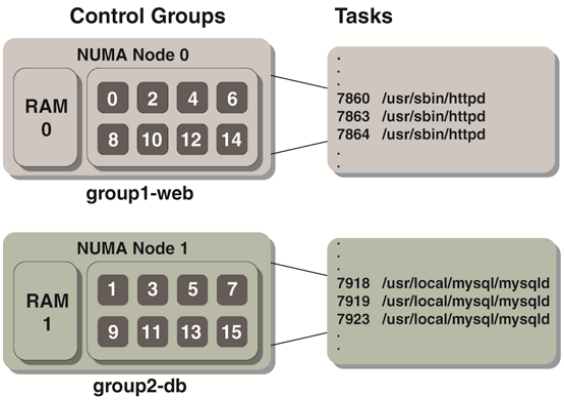
Ομάδα : **b04**

Ον/μο **: Βαβουλιώτης Γεώργιος**

Α.Μ. **: 03112083**

Εξάμηνο : 8

Για την άσκηση αυτή υποθέτουμε οτι έχουμε ενα data center το οποίο προσφέρει υπηρεσίες φιλοξενίας εφαρμογών. Υποθέτουμε οτι η απαίτηση που έχει μια εφαρμογή μετράται σε χιλιοστά της υπολογιστικής ισχύος ενός επεξεργαστή.
Όσο αφορά τις εφαρμογές, υπάρχει ενας διαχωρισμός σε ελαστικές και ανελαστικές εφαρμογές. Οι ανελαστικές εφαρμογές καταναλώνουν συνεχώς το σύνολο το χιλιοστών που τους αντιστοιχούν. Οι ελαστικές εφαρμογές δεν επηρεάζονται απο τις αυξομειώσεις της επεργαστικής ισχύος που λαμβάνουν κάθε στιγμή, αλλά πρέπει σε βάθος χρόνου να λάβουν όσο χρειάζονται.
Σκοπός μας στην άσκηση αυτή είναι η δημιουργία προγραμμάτων τα οποία με την χρήση των cgroups θα καταφέρουν να εξασφαλίσουν το σωστό διαμερισμό της υπολογιστικής ισχύος στις διάφορες εφαρμογές που έχει το data center.



Μας δίνεται ο δαίμων cgmon, ο οποίος αυτό που κάνει είναι να εκκινεί και να παρακολουθεί εφαρμογές αλλά και να επιβάλλει τα ελάχιστα εξασφαλισμένα χιλιοστά επεξεργαστικής ισχύος στη κάθε εφαρμογή. Για τους λόγους της άσκησης ο δαίμων cgmon αναθέτει τις παραπάνω δουλειές σε δυο άλλα προγράμματα τα οποία ονομάζονται cgmon-policy και cgmon-limit, τα οποία είναι ουσιαστικά αυτό το οποίο καλόυμαστε να υλοποιήσουμε.

Εξήγηση Λειτουργίας : Απο τη στιγμή μου ενεργοποιήσουμε τον δαίμονα τότε κάθε φορά που μια εφαρμογή δημιουργείται ή τερματίζεται τότε αυτός εκτελεί το cgmon-policy το οποίο έχει ως δουλεία να ‘παράγει’ όλες τις ρυθμίσεις για τα cgroup και να αποφανθεί αν οι απαιτήσεις της εφαρμογής μπορούν να καλυφθούν ή οχι(αν δεν μπορούν να καλυφθούν τότε εμφανίζεται κατάλληλο μήνυμα). Στη συνέχεια καλείται το cgmon-limit και το μόνο που κάνει είναι να εφαρμόζει ότι αποφάσισε το cgmon-policy. Στη πράξη αυτό που καλούμαστε να κάνουμε είναι να υλοποιήσουμε την παραπάνω ιδέα, δηλαδή να γράψουμε τα προγράμματα cgmon-policy και cgmon-limit ώστε όσες εφαρμογές επιτρέπουμε να μπουν στο data center να είναι εξασφαλισμένες οτι θα λάβουν τα ποσά υπολογιστικής ισχύς που τους έχουμε ‘υποσχεθεί’, να μην μπαίνουν εφαρμογές για τις οποίες δεν μπορούμε να εξασφαλίσουμε τους ελάχιστους απαιτούμενους πόρους και θα πρέπει να εξασφαλίσουμε πως αν υπάρχουν και τα δυο είδη εφαρμογών(ελαστικές, ανελαστικές) τότε κανένας κόμβος δεν θα είναι αδρανής αλλά όλη η διαθέσιμη υπολογιστική ισχύς θα διαμοιράζεται με κατάλληλο τρόπο στις υπάρχουσες εφαρμογές.
Για να υλοποιηθεί η παραπάνω ιδέα, αφού μελέτησα τους δοσμένους κώδικες, αποφάσισα να υλοποιήσω τα cgmon-policy και cgmon-limit σε γλώσσα Python, αφού η υλοποίηση τους με την γλώσσα αυτή ήταν πολύ απλή και ο κώδικας μικρός. Επίσης θα πρέπει να τονιστεί οτι για τις ανάγκες της άσκησης θεωρήθηκε οτι έχουμε στη διάθεσή μας 2000 πόρους υπολογιστικής ισχύος και οτι οι εφαρμογές χωρίζονται ανάλογα με τις απαιτήσεις τους σε:

Name	cpu
default_min100	100
default_min1000	1000
default_min500	500
elastic	50
platinum	1000
silver	500

Κώδικας cgmon-policy

```
#!/usr/bin/env python

import sys,os,math

sum=0 #accumulator for totalscore
elastic_cnt=0 #accumulator for elastics
inptlen=0 #accumulator for input length
power=2000.0 #constant
input_array=[]#put the apps in array that have score <=2000
inputvalue = sys.stdin.read().split('\n')#take the input
inputvalue = inputvalue[:-1]

for each_line in inputvalue:
    splitted=each_line.split(':') #split the input
    int_value=int(splitted[3]) #take the score
    sum += int_value
    if sum <= power
        inptlen++
        input_array.append(splitted)
        if int_value == 50:
            elastic_cnt += 1 #increase the counter of elastics
    else
        sum -=int_value

if sum <= power
    print "score:"+format((2000-sum)/2000.0) #format with 2000
    renew_score=2000-sum

for splitted in input_array:
    cpu_need=int(splitted[3])
    if inptlen==1:
        cpu_shares_renew=1024 #give all the cpu
    elif renew_score>0:
        #take that you deserve
        val1=cpu_need+renew_score*(sum-cpu_need)
        val2=(inptlen-1)*sum
        #normalize with 1024
        val3=(val1/val2)*1024/2000
        cpu_shares_renew= int(floor(val3))
        if cpu_shares_renew<0:
            cpu_shares_renew=0
    else:
        #take that you deserve
        val1=cpu_need+renew_score*cpu_need/sum
        #normalize with 1024
        val2=val1*1024/2000
        cpu_shares_renew= int(floor(val2)) #take the floor make it integer
    print 'set_limit:{}:cpu_need.shares:{}'.format(splitted[1],cpu_shares_renew)
```

Το πρόγραμμα cgmon-policy λαμβάνει είσοδο της μορφής:
`policy:<application name>:cpu:<value>`
όπου:

- `application name` : το όνομα μιας εφαρμογής
- `value` : μια θετική σταθερά που δηλώνει τις απαιτήσεις σε χιλιοστά υπολογιστικής ισχύος.
- Τα υπόλοιπα είναι σταθερές.

Επεξήγηση : Ο παραπάνω κώδικας αρχικά διαβάζει την είσοδο γραμμή γραμμή και αθροίζει σε εναν αθροιστή όλα τα scores, δηλαδή τα χιλιοστά υλπολογιστικής ισχύος που απαιτούνται για τις εφαρμογές, με την προυπόθεση το συνολικο άθροισμα να μην ξεπερναέι τις 2000 μονάδες. Επίσης χρησιμοποιώ εναν ακόμα αθροιστή ο οποίος μετράει τις ελαστικές εφαρμογές(τις ξεχωρίζει απο το γεγονός ότι ζητούν 50 χιλιοστά υπολογιστικής ισχύος). Όλα οι εφαρμογές που μπορούν να εξυπηρετηθούν απο το data center μπαίνουν σε ενα πίνακα και στη συνέχεια αν το άθροισμα των απαιτήσεων των εφαρμογών είναι μικρότερο τις διαθέσιμης υπολογιστικής ισχύος τότε το κάνω normalize με το 2000 και το τυπώνω. Τέλος για κάθε κελί του παραπάνω πίνακα, παίρνουμε τα χιλιοστά της cpu που απαιτεί η εφαρμογή και κάνω τους κατάλληλους υπολογισμούς για να διαμοιρασω σωστά την υπολογιστική ισχύ, η οποία ενδεχομένως να μην δεσμεύεται απο τις απαιτήσεις των εφαρμογών κι εμείς έχουμε το δικαίωμα να την κάνουμε οτι θέλουμε. Αυτό που κάνουμε είναι να την μοιράζουμε την ισχύ αυτή στις εφαρμογές που έχουμε μέσα στο data center.

Κώδικας cgmon-limit

```
#!/usr/bin/env python

import sys
import os

#define the right paths for each mode
path_stdr="/sys/fs/cgroup/cpu/"
path_tasks="/tasks"
path_cpushares="/cpu.shares"

# take the input, split it and take each part and use it as you have to
for input in sys.stdin:
    splitted = line.split(':',6)
    if splitted[0]=="create":
        #create the right path
        path =path_stdr+splitted[1]+"/"+splitted[3]
        os.system("mkdir "+path)
    elif splitted[0]=="remove":
        #create the right path
        path =path_stdr+splitted[1]+"/"+splitted[3]
        os.system("rmdir "+path)
    elif splitted[0]=="add":
        #create the right path
        path =path_stdr+splitted[1]+"/"+splitted[3]+path_tasks
        os.system("echo "+splitted[4]+" > "+path )#pid=splitted[4]
    elif splitted[0]=="set_limit":
        #create the right path
        path =path_stdr+splitted[1]+"/"+splitted[3]+path_cpushares
        os.system("echo "+splitted[5]+" > "+path ) #value=splitted[5]
```

Επεξήγηση : Το cgmon_limit είναι υπεύθυνο για :

- δημιουργία ενός νέου cgroup
- διαγραφή ενός cgroup
- εισαγωγή μιας εφαρμογής σε ένα cgroup το οποίο υπάρχει ήδη
- ορισμό των χιλιοστών που μπορούν να πάρουν οι εφαρμογές σε κάθε cgroup

Ο κώδικας του cgmon_limit είναι πολύ απλός, αρχικά διαβάζουμε την είσοδο γραμμή γραμμή και την 'σπάμε' κατάλληλα. Έπειτα ανάλογα με την εντολή που έχει δωθεί κάνουμε μια απο τις παρακάτω ενέργειες:

- **create**: δημιουργεί ενα νέο cgroup στο path `/sys/fs/cgroup/cpu/monitor/'Select_App_Name'`, όπου το `Select_App_Name` εξαρτάται απο το όνομα που θέλουμε να δώσουμε στην εφαρμογή. Η εντολή που εκτελέιται είναι λοιπόν η :
  `mkdir -p /sys/fs/cgroup/cpu/monitor/'Select_App_Name'`
  όπου η παράμετρος p υπάρχει ώστε να μην δημιουργηθούν directories με το ίδιο όνομα.
- **remove**: διαγράφει ενα cgroup, στο οποίο όμως **δεν υπάρχει μέσα καμία ενεργή διεργασία**, αν υπάρχει τότε αποτυγχάνει. Η εντολή που εκτελέιται είναι λοιπόν η :
  `rmdir -p /sys/fs/cgroup/cpu/monitor/'Select_App_Name'`
- **add**: προσθέτει μια εφαρμογή μέσα σε ένα ήδη υπάρχων cgroup γράφοντας το pid της στο αρχείο tasks του cgroup, το οποίο υπάρχει στο directory `/sys/fs/cgroup/cpu/monitor/'Select_App_Name'/` . Η εντολή θα είναι της μορφής :
  `echo (pid) > tasks`
- **set_limit**: καθορίζει τα χιλιοστά υπολογιστικής ισχύος που θα πάρει κάθε cgroup, γράφοντας με στο αρχείο cpu.shares του cgroup που μας ενδιαφέρει το νούμερο που μας δόθηκε από το policy. Η εντολή είναι ανάλογη με την παραπάνω γι'αυτό και αμελείται.

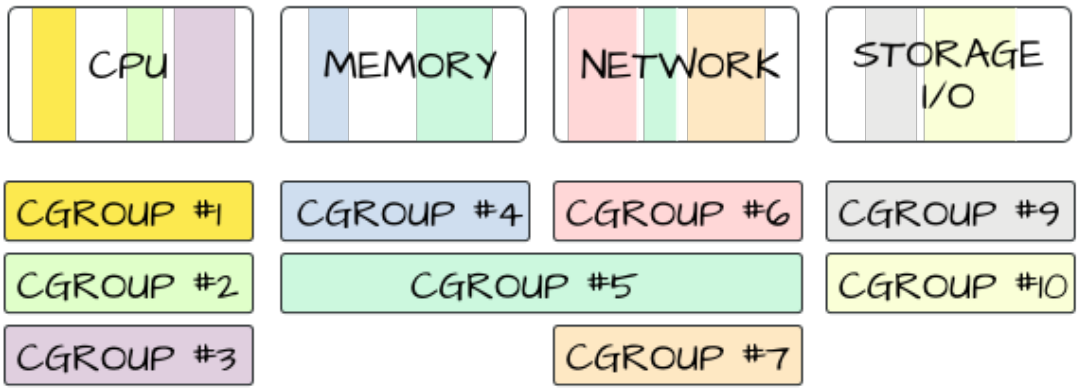
<u>Παρατηρήσεις Υλοποίησης</u> :

1. Για να τρέξει η άσκηση αυτή μετά την υλοποίηση των προγραμμάτων σε γλώσσα python έπρεπε να αλλάξουμε και τα δικαιώματα των source αρχείων για να είναι δυνατή η εκτέλεση τους απο τον δαίμονα(εντολή chmod). Αν κάτι τέτοιο δεν γινόταν όταν εκτελούσα την ./cgmon_demo.sh έπαιρνα μηνύματα της μορφής <<*Permission denied*>>.
2. Στους παραπάνω κώδικες θα συναντήσετε αρκετές φορές την εντολή os.system(*parameter*). Ο λόγος ύπαρξης της είναι η εκτέλεση εντολών bash και μου είναι πολύ χρήσιμη όπως φαίνεται και στους κώδικες.
3. Θα μπορούσα να κάνω τον κώδικά μου να είναι πιο αποδοτικός όσο αφορά τον διαμοιρασμό των πόρων που είναι διαθέσιμοι κάθε στιγμή. Η ιδέα είναι να μεταχειριστώ κατάλληλα τις ελαστικές εφαρμογές και συγκεκριμένα να υλοποιήσω

την παρακάτω ιδέα :

1. Χρησιμοποιώ μια ουρά στην οποία θα βάζω τις ελαστικές εφαρμογές.
2. Αν υπάρχει διαθέσιμοι υπολογιστική ισχύς τότε εκτελούνται τη στιγμή εκείνη. Μποούμε να εφαρμόσουμε scheduling στις ελαστικές εφαρμογές ώστε να καταφέρουμε να μην μείνει ανικανοποίητη κάποια η οποία έχει την δυνατότητα να εκτελεστεί.
3. Αν δεν υπάρχει διαθέσιμη υπολογιστική ισχύς τότε οι ελαστικές μένουν στην ουρά.
4. Αν κάποια εφαρμογή φύγει και ελευθερώσε πόρους ελέγχουμε αν μπορεί να ικανοποιηθεί κάποια-ες απο τις ελαστικές και αν αυτό μπορεί να γίνει προφανώς θα πρέπει να την βγάλω απο την ουρά.

Με την ιδέα αυτή θα γίνει καλύτερος διαμερισμός των πόρων αφού έτσι εγγυόμαστε ότι κάθε ανελαστική θα εξυπηρετηθεί και οι ελαστικές θα μπούν μόνο όταν χωράνε και δεν κλέβουν χρόνο απο τις ανελαστικές.. Βέβαια ο σωστότερος τρόπος ποικίλλει ανάλογα με τις ανάγκες μας και το είδος των εφαρμογών που έχουμε να διαχειριστούμε.



Στο σημείο αυτό κρίνεται αναγκαίο να σας δείξω κάποια στιγμιότυπα εκτέλεσης του δαίμονα, για να γίνει σαφής η σωστή λειτουργία των προγραμμάτων:

1. Τρέχω το `cgmon_demo.sh` το οποίο γεννάει κάποιες εφαρμογές και στο σταματάω(Ctrl-C) λίγο πριν το τέλος για να μπορέσω να δω αν έχουν όντως γεννηθεί οι εφαρμογές που πρέπει. Με την εντολή top βλέπω τις εφαρμογές:

```
top - 20:13:01 up 10 days, 23:49,  3 users,  load average: 6.72, 7.22, 5.30
Tasks:  87 total,   9 running,  78 sleeping,   0 stopped,   0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2058464 total,  1749424 used,   309040 free,    53824 buffers
KiB Swap:        0 total,        0 used,        0 free.  1530824 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
20749 root      20   0    7172     92      0 R  52.6  0.0   0:07.76 stress
20748 root      20   0    7172     92      0 R  52.2  0.0   0:07.73 stress
20774 root      20   0    7172     88      0 R  29.3  0.0   0:04.18 stress
20775 root      20   0    7172     88      0 R  29.3  0.0   0:04.32 stress
20793 root      20   0    7172     92      0 R   9.0  0.0   0:01.13 stress
20795 root      20   0    7172     88      0 R   9.0  0.0   0:01.11 stress
20796 root      20   0    7172     88      0 R   9.0  0.0   0:01.11 stress
20794 root      20   0    7172     92      0 R   8.6  0.0   0:01.12 stress
20797 root      20   0   92160   5992   4960 S   1.0  0.3   0:00.06 sshd
    1 root      20   0   28696   4668   2968 S   0.0  0.2   0:17.63 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.14 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   1:54.36 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:05.25 kworker/u4:0
    7 root      20   0       0      0      0 S   0.0  0.0   0:57.05 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.36 migration/0
   10 root      rt   0       0      0      0 S   0.0  0.0   0:04.42 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:03.58 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.24 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   1:10.68 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   16 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 khelper
   17 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   18 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.16 khungtaskd
   20 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   21 root      25   5       0      0      0 S   0.0  0.0   0:00.00 ksmd
   22 root      39  19       0      0      0 S   0.0  0.0   0:00.00 khugepaged
   23 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 crypto
   24 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
   25 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   26 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kblockd
   29 root      20   0       0      0      0 S   0.0  0.0   0:00.07 kswapd0
   30 root      20   0       0      0      0 S   0.0  0.0   0:00.00 fsnotify_mark
   36 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kthrotld
   37 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 ipv6_addrconf
   38 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 deferwq
   39 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kworker/u4:1
   74 root      20   0       0      0      0 S   0.0  0.0   0:00.00 khubd
   75 root       0 -20       0      0      0 S   0.0  0.0   0:00.08 ata_sff
   76 root      20   0       0      0      0 S   0.0  0.0   0:00.00 scsi_eh_0
   77 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 scsi_tmf_0
   78 root      20   0       0      0      0 S   0.0  0.0   0:00.00 scsi_eh_1
   79 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 scsi_tmf_1
   86 root       0 -20       0      0      0 S   0.0  0.0   0:00.53 kworker/1:1H
  103 root       0 -20       0      0      0 S   0.0  0.0   0:01.35 kworker/0:1H
  106 root      20   0       0      0      0 S   0.0  0.0   0:06.11 jbd2/vda1-8
  107 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 ext4-rsv-conver
```

Παρατηρήσεις :

- Το `cgmon_demo.sh` δημιουργεί μια platinum, μια silver και δυο elastic εφαρμογές και παρατηρούμε οτι αυτό το οποιο γίνεται είναι να παίρνει κάθε εφαρμογή την υπολογιστική ισχύ την οποία τις είχαμε υποσχεθεί, κάτι το οποίο γίνεται εμφανές απο την στήλη %CPU της παραπάνω εικόνας. Ωστόσο παρατηρούμε οτι όλες οι εφαρμογές παίρνουν κάτι παραπάνω απ' όσο τους είχαμε υποσχεθεί, γεγονός το οποίο είναι λογικό αφού αν αθροίσουμε τις απαιτήσεις των εφαρμογών αυτών προκύπτει ο αριθμός 1000+500+2*50 = 1600 < 2000, δηλαδή επειδή δεν υπάρχει άλλη εφαρμογή στο data center η υπολογιστική ισχύς που μας απομένει, την διαμοιράζουμε στις υπάρχουσες εφαρμογές σύμφωνα με την σχέση η οποία υπάρχει στον κώδικα του cgmon-policy και καθορίζει τον διαμερισμό των ανεκμετάλλευτων υπολογιστικών πόρων.
- Η απεικόνιση των εφαρμογών έγινε με χρήση την εντολής top.
- Ο λόγος για τον οποίο δεν άφησα το `cgmon_demo.sh` να τερματίσει είναι οτι στο τέλος του υπάρχει η εντολή *pkill -f stress* η οποία σκοτώνει όλες τις stress εφαρμογές γεγονός το οποίο δεν θα μου επέτρεπε να δω με την εντολή top τις εφαρμογές που είχαν δημιουργηθεί.

2. Εκτελώ την εντολή *pkill -f stress* ώστε να 'σκοτώσω ' τις εφαρμογές που έβαλε το `cgmon_demo.sh` και στη συνέχεια πάω στο directory */root/cgmon* και πατάω την εντολή cgmon η οποία ενεργοποιεί τον δαίμονα και πλέον θα μπορώ να βάζω χειροκίνητα εφαρμογές. Για να μπορέσω να δημιουργώ εφαρμογές χρησιμοποιώ την εντολή :

```
app spawn -p SELECT1 -e " stress -c 2" -n SELECT2
```

όπου :

- **SELECT1** είναι το είδος της εφαρμογής(platinum, silver,…).
- **SELECT2** είναι το όνομα της εφαρμογής, το οποίο είναι ότι έγω επιθυμώ.

Άρα εκτελώ της εντολές που φαίνονται στο παρακάτω screenshoot :

```
root@snf-699535:~/cgmon# cgmon
(cgmon) app spawn -p platinum -e "stress -c 2" -n MYAPP1
(cgmon) app spawn -p silver -e "stress -c 2" -n MYAPP2
(cgmon) app spawn -p silver -e "stress -c 2" -n MYAPP3
(cgmon)
```

Με χρήση της εντολής top βλέπω πως έγινε ο διαμοιρασμός των πόρων στις εφαρμογές :

```
top - 21:52:21 up 11 days,  1:28,  2 users,  load average: 3.38, 0.94, 0.42
Tasks:  81 total,   7 running,  74 sleeping,   0 stopped,   0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2058464 total,  1757588 used,   300876 free,    53836 buffers
KiB Swap:        0 total,        0 used,        0 free.  1531228 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
21494 root      20   0    7172     88      0 R  49.9  0.0   0:35.58 stress
21495 root      20   0    7172     88      0 R  49.9  0.0   0:35.65 stress
21509 root      20   0    7172     88      0 R  25.0  0.0   0:13.15 stress
21510 root      20   0    7172     88      0 R  25.0  0.0   0:13.18 stress
21525 root      20   0    7172     88      0 R  25.0  0.0   0:10.14 stress
21526 root      20   0    7172     88      0 R  25.0  0.0   0:10.14 stress
    1 root      20   0   28696   4668   2968 S   0.0  0.2   0:17.83 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.14 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   2:02.99 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:05.32 kworker/u4:0
    7 root      20   0       0      0      0 S   0.0  0.0   1:01.24 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.37 migration/0
   10 root      rt   0       0      0      0 S   0.0  0.0   0:04.49 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:03.66 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.25 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   1:15.99 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   16 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 khelper
   17 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   18 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.17 khungtaskd
   20 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   21 root      25   5       0      0      0 S   0.0  0.0   0:00.00 ksmd
   22 root      39  19       0      0      0 S   0.0  0.0   0:00.00 khugepaged
   23 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 crypto
   24 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
   25 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   26 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kblockd
   29 root      20   0       0      0      0 S   0.0  0.0   0:00.07 kswapd0
   30 root      20   0       0      0      0 S   0.0  0.0   0:00.00 fsnotify_mark
   36 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kthrotld
   37 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 ipv6_addrconf
   38 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 deferwq
   39 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kworker/u4:1
   74 root      20   0       0      0      0 S   0.0  0.0   0:00.00 khubd
   75 root       0 -20       0      0      0 S   0.0  0.0   0:00.08 ata_sff
   76 root      20   0       0      0      0 S   0.0  0.0   0:00.00 scsi_eh_0
   77 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 scsi_tmf_0
   78 root      20   0       0      0      0 S   0.0  0.0   0:00.00 scsi_eh_1
   79 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 scsi_tmf_1
   86 root       0 -20       0      0      0 S   0.0  0.0   0:00.54 kworker/1:1H
  103 root       0 -20       0      0      0 S   0.0  0.0   0:01.37 kworker/0:1H
  106 root      20   0       0      0      0 S   0.0  0.0   0:06.21 jbd2/vda1-8
  107 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 ext4-rsv-conver
  137 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kauditd
  150 root      20   0   29924   3548   3260 S   0.0  0.2   1:10.47 systemd-journal
  152 root      20   0   40748   3112   2636 S   0.0  0.2   0:00.23 systemd-udevd
```

Σχολιασμός : Παρατηρώ ότι ο διαμοιρασμός των πόρων έγινε σωστά αφού οι διαθέσιμοι πόροι μου είναι 2000 αρχικά και η πρώτη εφαρμογή ζητά 1000 αφού είναι platinum και οι δυο επόμενες ζητούν απο 500 αφού είναι silver. Άρα οπώς είναι λογικό η platinum θα καταλάβει το 50% της CPU και οι άλλες δυο απο 25%. Πλέον έχουν διαμοιραστεί και οι 2000 πόροι που είχα στη διάθεση μου και αν κάποια εφαρμογή ζητήσει να μπει τότε δεν θα τα καταφέρει και θα εμφανιστεί κατάλληλο μήνυμα. Συγκεκριμένα αν εκτελέσω τις εντολές που φαίνονται παρακάτω παίρνω κατάλληλα μηνύματα έλλειψης πόρων, λόγω αδυναμίας του data center να καλύψει τις ανάγκες των εφαρμογών αυτών :

```
root@snf-699535:~/cgmon# cgmon
(cgmon) app spawn -p platinum -e "stress -c 2" -n MYAPP1
(cgmon) app spawn -p silver -e "stress -c 2" -n MYAPP2
(cgmon) app spawn -p silver -e "stress -c 2" -n MYAPP3
(cgmon)
(cgmon) app spawn -p silver -e "stress -c 2" -n MYAPP4
ERROR::Server returned: 'Resources 'CPU' returned negative scores'
(cgmon) app spawn -p platinum  -e "stress -c 2" -n MYAPP4
ERROR::Server returned: 'Resources 'CPU' returned negative scores'
(cgmon) app spawn -p elastic  -e "stress -c 2" -n MYAPP4
ERROR::Server returned: 'Resources 'CPU' returned negative scores'
(cgmon)
```

3. Οι δυνατότητες που μας δίνει η άσκηση αυτή είναι πολλές γεγονός. Θα μπορούσαμε να έχουμε ανοιχτά δυο τερματικά και να βάλουμε το ένα σαν εφαρμογή σε ένα cgroup. Αυτό μπορεί να γίνει εκτελώντας την παρακάτω εντολή :

```
echo $$  >  tasks
```

Για να γίνει αυτό θα πρέπει να έχουμε πάει στο directory `/sys/fs/cgroup/cpu/monitor/APPNAME` όπού `APPNAME` είναι το όνομα μιας εφαρμογής που έχουμε δημιουργήσει κατά τα γνωστά και tasks είναι ενα αρχείο που περιέχει τα pid των διεργασιών της κάθε εφαρμογής.

```
root@snf-699535:/sys/fs/cgroup/cpu/monitor/MYAPP1# echo $$ > tasks
```

Παρατηρώ οτι το αρχείο tasks, του οποίου το περιεχόμενο φαίνεται παρακάτω, έχει 4 pid μέσα και όχι 3 όπως είχε αρχικά, γεγονός το οποίο επιβεβαιώνει οτι αυτό που ισχυρίστηκα είναι ορθό.

```
19976
21628
21631
21632
~
~
```

4. Κάτι το οποίο είναι προφανές αλλά είναι καλό να το δείξω είναι το γεγονός οτι αν υπάρχει μόνο μια εφαρμογή τότε θα πάρει το 100% της CPU, γεγονός το οποίο φαίνεται παρακάτω :

```
top - 22:30:35 up 11 days,  2:07,  2 users,  load average: 0.68, 1.60, 2.34
Tasks:  75 total,   3 running,  72 sleeping,   0 stopped,   0 zombie
%Cpu(s):100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   2058464 total,  1757800 used,   300664 free,    53836 buffers
KiB Swap:        0 total,        0 used,        0 free.  1531404 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
21947 root      20   0    7172     88      0 R 100.0  0.0   0:07.34 stress
21948 root      20   0    7172     88      0 R  99.9  0.0   0:07.33 stress
20109 root      20   0       0      0      0 S   0.3  0.0   0:02.21 kworker/1:3
    1 root      20   0   28696   4668   2968 S   0.0  0.2   0:17.87 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.14 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   2:03.25 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:05.34 kworker/u4:0
    7 root      20   0       0      0      0 S   0.0  0.0   1:01.44 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:00.37 migration/0
   10 root      rt   0       0      0      0 S   0.0  0.0   0:04.50 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:03.67 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.25 migration/1
   13 root      20   0       0      0      0 S   0.0  0.0   1:16.60 ksoftirqd/1
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   16 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 khelper
   17 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   18 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.17 khungtaskd
   20 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   21 root      25   5       0      0      0 S   0.0  0.0   0:00.00 ksmd
   22 root      39  19       0      0      0 S   0.0  0.0   0:00.00 khugepaged
   23 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 crypto
   24 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
   25 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   26 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kblockd
   29 root      20   0       0      0      0 S   0.0  0.0   0:00.07 kswapd0
   30 root      20   0       0      0      0 S   0.0  0.0   0:00.00 fsnotify_mark
   36 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kthrotld
   37 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 ipv6_addrconf
   38 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 deferwq
   39 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kworker/u4:1
   74 root      20   0       0      0      0 S   0.0  0.0   0:00.00 khubd
   75 root       0 -20       0      0      0 S   0.0  0.0   0:00.08 ata_sff
   76 root      20   0       0      0      0 S   0.0  0.0   0:00.00 scsi_eh_0
   77 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 scsi_tmf_0
   78 root      20   0       0      0      0 S   0.0  0.0   0:00.00 scsi_eh_1
   79 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 scsi_tmf_1
   86 root       0 -20       0      0      0 S   0.0  0.0   0:00.54 kworker/1:1H
  103 root       0 -20       0      0      0 S   0.0  0.0   0:01.38 kworker/0:1H
  106 root      20   0       0      0      0 S   0.0  0.0   0:06.23 jbd2/vda1-8
  107 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 ext4-rsv-conver
  137 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kauditd
  150 root      20   0   29924   3932   3644 S   0.0  0.2   1:10.73 systemd-journal
  152 root      20   0   40748   3112   2636 S   0.0  0.2   0:00.23 systemd-udevd
  182 root      20   0       0      0      0 S   0.0  0.0   0:00.00 vballoon
  187 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kpsmoused
  377 root      20   0   25400   8600   1704 S   0.0  0.4   0:00.44 dhclient
```