Practica 4 – 1

Parte 1: Conceptos teóricos

1. Defina virtualización. Investigue cuál fue la primera implementación que se realizó.

   Virtualización:

   - Técnica que permite realizar “abstracción” de los recursos de la computadora.
   - Proceso que permite a una computadora compartir sus recursos de hardware con varios entornos separados de forma digital.
   - Capa abstracta que desacopla el hardware físico del Sistema.
   - Cada entorno virtualizado se ejecuta dentro de los recursos asignados.
   - Permite ocultar detalles técnicos a través de la encapsulación.
   - Permite que una computadora pueda realizar el trabajo de varias, a través de la compartición de recursos de un único dispositivo de hardware.
   - Permite cambiar entre diferentes sistemas operativos en el mismo servidor sin tener que reiniciar.

   La virtualización se introdujo por primera vez en el IBM System/370 en 1972, para su uso con VM/370, el primer sistema operativo de máquina virtual ofrecido por IBM como producto oficial.

   VM/370 es un sistema operativo orientado a máquinas virtuales que se lanzó en el año 1972 por IBM. El SO corría en entornos Mainframe con arquitectura System/370, System/390 y más avanzados en zSeries. Utilizaba como núcleo a VMCP (Virtual Machine control program), lo que permite la ejecución de las VM y el control del hardware

Funcionamiento:

- En VM/370, hay una máquina CMS (Conversational Monitor System) para cada usuario, con “la ilusión” del hardware completo.
- Una aplicación sobre CMS hace una system call, y la “atrapa” (es un trap) CMS.
    - Ejemplo: CMS emite instrucciones de I/O que son traps a VM/370

2. ¿Qué diferencia existe entre virtualización y emulación?

Virtualización: particionar un procesador físico en distintos contextos, donde cada uno de ellos corre sobre el mismo procesador.

- Los SO guest deben ejecutar la misma arquitectura de hardware sobre la que corren.
- No requiere que los guest se modifiquen.
- Técnica más utilizada.
- El VMM analiza el flujo de ejecución.
- Los bloques que contienen instrucciones sensibles son modificados.
- Los bloques con instrucciones inocuas se ejecutan directamente en el hardware.

Emulación: provee toda la funcionalidad del procesador deseado a través de software.

- Se reescribe el conjunto completo de instrucciones.
- Se puede emular un procesador sobre otro tipo de procesador.
- Lenta.

Virtualización	***Emulación***
Proceso de creación de una versión virtual de algo como un sistema operativo, un servidor, un	Proceso de simulación de otro tipo de dispositivo.

Virtualización	***Emulación***
dispositivo de almacenamiento, etc.	
Generalmente aumenta la agilidad de TI, aumenta la escalabilidad, mejora la flexibilidad, ahorra costos, etc.	Generalmente aumenta la satisfacción del cliente, instala aceleración en el sitio del cliente que minimiza el impacto en la producción, aumenta la confianza, etc.
Simplemente nos permite y nos ayuda a utilizar la capacidad total de las máquinas físicas distribuyendo sus capacidades entre diferentes entornos o usuarios.	Simplemente nos permite y nos ayuda a ejecutar software desde un dispositivo totalmente diferente en nuestra computadora.
Su principal objetivo es centralizar las tareas administrativas, lo que a su vez mejora la escalabilidad y las cargas de trabajo.	Su objetivo principal es permitir que el subsistema presente la misma interfaz y características que otro.

Virtualización	***Emulación***
Ejecuta código directamente con diferentes conjuntos de dominios en el lenguaje de uso.	Utiliza un intérprete que traduce y convierte el código fuente a un formato legible del sistema host para que pueda procesarse más.
Crea un ambiente aislado.	Reproduce el comportamiento del mismo hardware, incluidas peculiaridades, errores, etc.
Por lo general, accede al hardware directamente sin necesidad de un puente de software.	Por lo general, requiere un puente de software para acceder al hardware.
Es menos rentable, es decir, más costoso en comparación con la emulación.	Es más rentable en comparación con la virtualización.
Proporciona soluciones de copia de seguridad mejores y eficaces en comparación con la emulación.	Proporciona una solución de copia de seguridad menos eficaz en comparación con la virtualización.
Hace uso y depende de la CPU.	No utiliza CPU.

Virtualización	***Emulación***
Normalmente es más rápido que la emulación.	Es innecesariamente más lento que la virtualización.

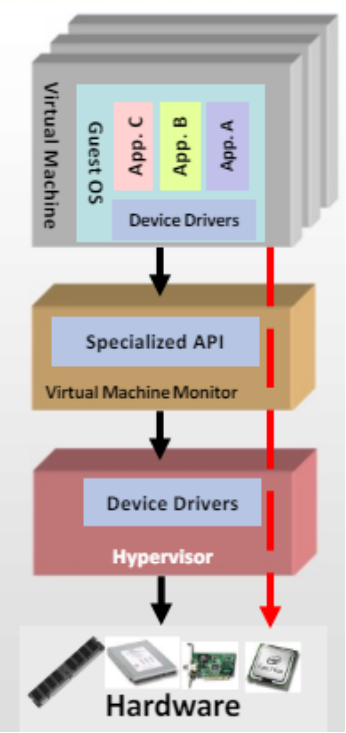
3. Investigue el concepto de hypervisor y responda:
   a) ¿Qué es un hypervisor?

      Componente de software que administra varias máquinas virtuales en una computadora. Separa las “aplicaciones/SO” del hardware subyacente

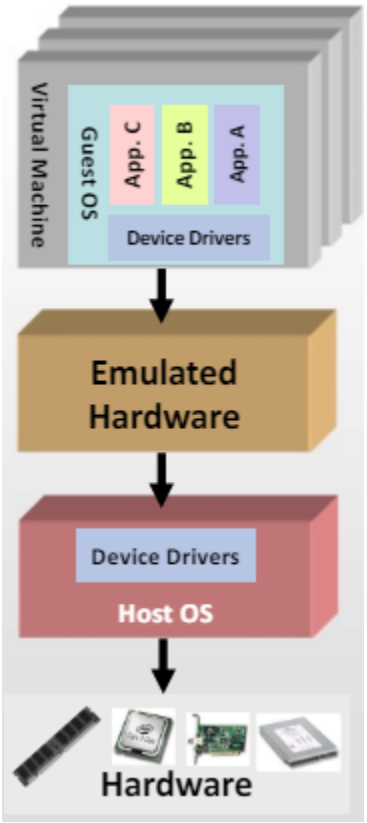
   b) ¿Qué beneficios traen los hypervisors? ¿Cómo se clasifican?
      Garantiza que cada máquina virtual reciba los recursos asignados y no interfiera con el funcionamiento de otras máquinas virtuales.

      <u>Clasificación:</u>

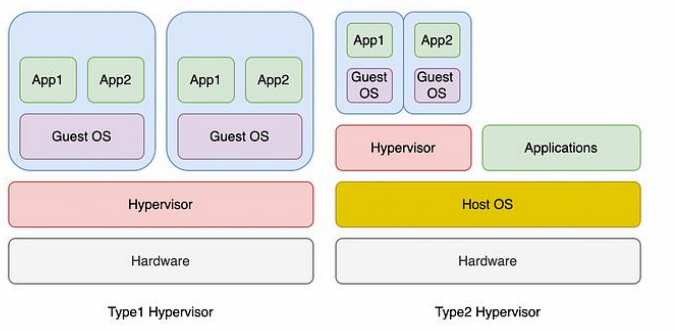
- Hipervisor de tipo 1



  - Hipervisor bare metal
  - En vez de instalarse en el sistema operativo, se instala de forma directa en el hardware de la computadora. Se ejecuta sobre HW.
  - Tienen un mejor rendimiento y se utilizan con frecuencia para las aplicaciones empresariales.
  - Hipervisor se ejecuta en modo kernel y el SO guest en modo kernel “virtual” (modo usuario). Es decir, VM se ejecuta como un proceso de usuario en modo usuario.
  - SO guest no requiere ser modificado.
  - Existen un modo kernel virtual y modo usuario virtual.
  - Siempre que la VM ejecuta una instrucción sensible, se produce una trap que procesa el hypervisor. Algunos introducen extensiones para evitar traducir todas las instrucciones
    - Asistencia de Hardware Siempre
- Hipervisor de tipo 2



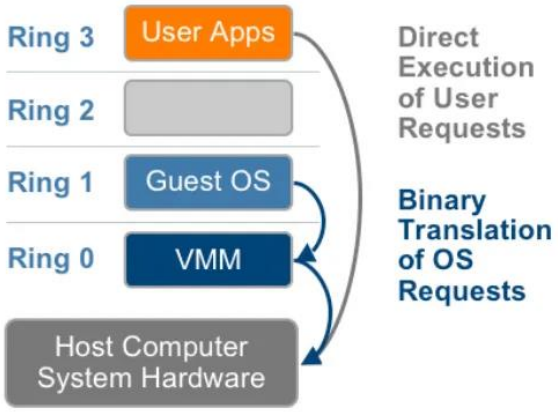
  - Hipervisor alojado
  - Está instalado en un sistema operativo. Se ejecuta como un programa de usuario en un SO host.
  - Interpreta un subconjunto de las instrucciones de hardware de la máquina sobre la que corre.
  - Son adecuados para el usuario final.
  - Arriba de él, están los SO guests.
  - El hardware que se mapea a los SO guest se emula.
  - El SO host es quién se ejecuta sobre el HW



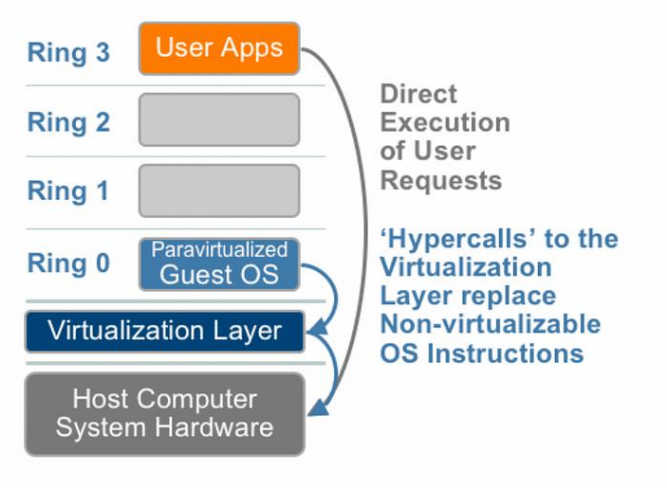
4. ¿Qué es la full virtualization? ¿Y la virtualización asistida por hardware?

   Existen tres técnicas alternativas para manejar instrucciones confidenciales y privilegiadas para virtualizar la CPU en la arquitectura:

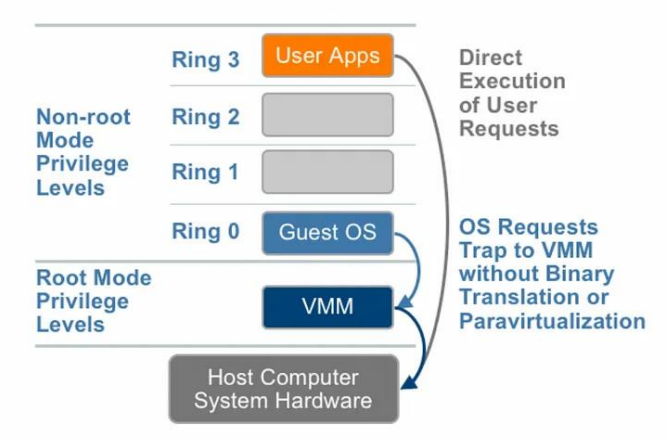
   - Virtualización completa mediante traducción binaria
     - Virtualización de cualquier SO usando traducción binaria y ejecución directa.
     - Traducción del código del kernel para reemplazar instrucciones no virtualizables con nuevas secuencias.
     - La capa de virtualización abstrae completamente el sistema operativo invitado del hardware subyacente.
     - El SO guest no requiere modificaciones y no sabe que se está virtualizando.
     - La virtualización completa no necesita asistencia de hardware ni del SO.
     - El hipervisor traduce y almacena en caché las instrucciones del SO en tiempo real. El hipervisor simula completamente el hardware subyacente.
     - Usa una combinación de ejecución directa y traducción binaria permitiendo la ejecución directa de instrucciones de CPU no sensibles, mientras que las instrucciones de CPU sensibles se traducen sobre la marcha.
       - Para mejorar el rendimiento, el hipervisor mantiene una caché de las instrucciones recientemente traducidas.
       - Ejecución directa del código a nivel de usuario en el procesador para alto rendimiento.
     - Ofrece mejor aislamiento, seguridad, y facilita migración y portabilidad del sistema operativo invitado.



- Virtualización o paravirtualización asistida por SO
    - Implica la comunicación entre SO guest y el hipervisor para mejorar rendimiento y eficiencia.
    - Modificación del kernel del SO para reemplazar instrucciones no virtualizables con hiperllamadas directas al hipervisor.
    - El hipervisor no simula el hardware subyacente, si no que proporciona hiperllamadas. El SO guest usa las hiperllamadas para ejecutar instrucciones sensibles de la CPU.
    - No es tan portátil como la virtualización completa ya que requiere modificaciones en el SO guest y no soporta SO no modificados.
    - Proporciona un mejor rendimiento porque el SO guest es consciente de que está siendo virtualizado.
    - Puede introducir problemas de soporte y mantenibilidad debido a las modificaciones profundas del kernel.
    - Crear soporte de traducción binaria para virtualización completa es difícil, pero modificar el SO para paravirtualización es relativamente fácil.
    - Los cambios en el SO guest para permitir la paravirtualización son mínimos y no intrusivos



- Virtualización asistida por hardware (primera generación)
    - El hardware subyacente proporciona instrucciones especiales de la CPU para ayudar a la virtualización.
    - Altamente portátil, ya que el hipervisor puede ejecutar un sistema operativo invitado sin modificaciones.
    - Hace que la implementación del hipervisor sea menos compleja y más fácil de mantener.
    - Los procesadores Intel-VT de Intel y AMD-V de AMD proporcionan instrucciones de virtualización de CPU que los proveedores de software utilizan para implementar la virtualización asistida por hardware.



5. ¿Qué implica la técnica binary translation? ¿Y trap-and-emulate?

- Binary translation implica la traducción dinámica de las instrucciones del SO guest que no pueden ejecutarse directamente en el hardware del host. Estas son interceptadas por el hipervisor y traducidas en tiempo real a un conjunto equivalentes que pueden ejecutarse de manera segura en el hardware del host.
    - Puede introducir una sobrecarga significativa debido al proceso de traducción en tiempo real.
- Trap-and-emulate cuando el SO guest ejecuta una instrucción privilegiada, el hardware del host (con soporte de virtualización) genera un trap que transfiere el control al hipervisor, este emula la instrucción privilegiada de manera que simula el comportamiento que el SO guest espera, pero de una forma controlada y segura. Una vez que la instrucción ha sido emulada, el control se devuelve al sistema operativo invitado, permitiéndole continuar su ejecución.
    - Depende del soporte de hardware para capturar y emular instrucciones privilegiadas

6. Investigue el concepto de paravirtualización y responda:

    a) ¿Qué es la paravirtualización?

    Punto 4.

    b) Mencione algún sistema que implemente paravirtualización.

    El proyecto de código abierto Xen que virtualiza el procesador y la memoria con un kernel de Linux modificado y las E/S con controladores de dispositivos personalizados.

    c) ¿Qué beneficios trae con respecto al resto de los modos de virtualización?

    Mejor rendimiento.

7. Investigue sobre containers y responda:

    a) ¿Qué son?

Tecnología liviana de virtualización (lightweight virtualization) a nivel de sistema operativo que permite ejecutar múltiples sistemas aislados (conjuntos de procesos) en un único host. Las características del kernel del SO (por ejemplo, espacios de nombres y cgroups de Linux, silos de Windows y objetos de trabajo) se pueden usar para aislar procesos y controlar la cantidad de CPU, memoria y disco a los que pueden acceder esos procesos.

b) ¿Dependen del hardware subyacente?

No, no dependen, ya que en lugar de virtualizar el hardware subyacente virtualizan el sistema operativo de manera que cada contenedor individual contiene solo la aplicación y sus bibliotecas y dependencias.

c) ¿Qué lo diferencia por sobre el resto de las tecnologías estudiadas?

Son ligeros, rápidos y portátiles porque, a diferencia de una máquina virtual, no necesitan incluir un SO invitado en cada instancia y, en su lugar, simplemente pueden usar las características y recursos del SO host.

d) Investigue qué funcionalidades son necesarias para poder implementar containers.
    - Namespace
    - Cgroups
    - Union File Systems
    - Kernel Capabilities:
        - Permite limitar las capacidades de root dentro de los containers para mejorar la seguridad.
        - Reducción del conjunto de capacidades que un container puede utilizar, limitando las acciones que pueden realizar incluso si se compromete el container.
    - Container Runtime:
        - Ej. Docker.
    - Image Registry:
        - Un repositorio donde se almacenan y distribuyen las imágenes de containers.

- Networking:
  - Bridge Networks: Conecta containers en un host a través de una red puente.
  - Overlay Networks: Permite que containers en diferentes hosts se comuniquen a través de una red virtual.
  - Host Networks: Los containers comparten la pila de red del host.
- Security Enhancements:
  - Seccomp: Permite filtrar las llamadas al sistema que puede hacer un container, restringiendo su interacción con el kernel.
  - AppArmor y SELinux: Sistemas de control de acceso que pueden ser usados para imponer políticas de seguridad sobre containers.
  - Linux Capabilities: Limitación granular de permisos de root dentro de containers.
- Storage Drivers:
  - Soporte para diferentes drivers de almacenamiento para gestionar la persistencia de datos de los containers.

Parte 2: chroot, Control Groups y Namespaces

Chroot

1. ¿Qué es el comando chroot? ¿Cuál es su finalidad?

   Es una forma de aislar aplicaciones del resto del sistema cambiando el directorio raíz aparente de un proceso afectándolo a él y a sus procesos hijos.

2. Crear un subdirectorio llamado sobash dentro del directorio root. Intente ejecutar el comando chroot /root/sobash. ¿Cuál es el resultado? ¿Por qué se obtiene ese resultado?

```
root@so2022:~/sobash# chroot /root/sobash
chroot: failed to run command '/bin/bash': No such file or directory
root@so2022:~/sobash#
```

No se encuentra el archivo /bin/bash necesario para ejecutar el comando bash.

3. Copiar en el directorio anterior todas las librerías que necesita el comando bash. Para obtener esta información ejecutar el comando ldd /bin/bash. ¿Es necesario copiar la librería linux-vdso.so.1? ¿Por qué? Dentro del directorio anterior crear las carpetas donde va el comando bash y las librerías necesarias. Probar nuevamente. ¿Qué sucede ahora?

   mkdir bin
   mkdir lib
   mkdir -p "/root/sobash/lib/x86_64-linux-gnu/" "/root/sobash/lib64/" "/root/sobash/bin/"
   cp "/lib/x86_64-linux-gnu/libtinfo.so.6" "/root/sobash/lib/x86_64-linux-gnu/"
   cp "/lib/x86_64-linux-gnu/libdl.so.2" "/root/sobash/lib/x86_64-linux-gnu/"
   cp "/lib/x86_64-linux-gnu/libc.so.6" "/root/sobash/lib/x86_64-linux-gnu/"
   cp "/lib64/ld-linux-x86-64.so.2" "/root/sobash/lib/x86_64-linux-gnu/"
   cp "/lib64/ld-linux-x86-64.so.2" "/root/sobash/lib64/"
   cp "/bin/bash" "/root/sobash/bin/"

   Ahora si se ejecuta exitosamente. No es necesaria copiar la librería linux-vdso.so.1 porque no es un archivo estático que reside en el sistema de archivos como las otras librerías, sino que es gestionado dinámicamente por el kernel Linux. Es una biblioteca virtual proporcionada por el kernel de Linux y es cargada automáticamente por el sistema operativo en tiempo de ejecución.

4. ¿Puede ejecutar los comandos cd "directorio" o echo? ¿Y el comando ls? ¿A qué se debe esto?

   Si se puede ejecutar los comandos cd y echo, pero no ls debido a que ls es un ejecutable externo.

5. ¿Qué muestra el comando pwd? ¿A qué se debe esto?

   Muestra / . Esto se debe a que toma como directorio raíz el directorio donde se ejecutó chroot, creando una especie de “jail”, haciendo que no se pueda acceder ni ver archivos y comandos fuera del directorio.

6. Salir del entorno chroot usando exit

Control Groups

1. ¿Dónde se encuentran montados los cgroups? ¿Qué versiones están disponibles?

   Se encuentran montados en /sys/fs/cgroup/. Hay dos versiones disponibles, cgroups v1 y cgroups v2

2. ¿Existe algún controlador disponible en cgroups v2? ¿Cómo puede determinarlo?

   No, no existe ningún controlador disponible en cgroups v2, lo puedo determinar con el siguiente comando:
   cat /sys/fs/cgroup/unified/cgroup.controllers

3. Analice qué sucede si se remueve un controlador de cgroups v1 (por ej. Umount /sys/fs/cgroup/rdma).

```
root@so2022:~# ls /sys/fs/cgroup/rdma/
cgroup.clone_children  cgroup.sane_behavior  release_agent
cgroup.procs           notify_on_release     tasks
root@so2022:~# umount /sys/fs/cgroup/rdma
root@so2022:~# ls /sys/fs/cgroup/rdma/
```

   El SO elimina los archivos y carpetas que se crearon cuando se montó.

4. Crear dos cgroups dentro del subsistema cpu llamados cpualta y cpubaja. Controlar que se hayan creado tales directorios y ver si tienen algún contenido
   # mkdir /sys/fs/cgroup/cpu/"nombre_cgroup"

```
root@so2022:~# ls /sys/fs/cgroup/cpu/cpualta
cgroup.clone_children  cpuacct.usage_percpu_sys   cpu.shares
cgroup.procs           cpuacct.usage_percpu_user  cpu.stat
cpuacct.stat           cpuacct.usage_sys          notify_on_release
cpuacct.usage          cpuacct.usage_user         tasks
cpuacct.usage_all      cpu.cfs_period_us
cpuacct.usage_percpu   cpu.cfs_quota_us
root@so2022:~# ls /sys/fs/cgroup/cpu/cpubaja
cgroup.clone_children  cpuacct.usage_percpu_sys   cpu.shares
cgroup.procs           cpuacct.usage_percpu_user  cpu.stat
cpuacct.stat           cpuacct.usage_sys          notify_on_release
cpuacct.usage          cpuacct.usage_user         tasks
cpuacct.usage_all      cpu.cfs_period_us
cpuacct.usage_percpu   cpu.cfs_quota_us
```

5. En base a lo realizado, ¿qué versión de cgroup se está utilizando?

   Se está usando la versión 1. Si quiero usar la 2 debería trabajar sobre /sys/fs/cgroup/unified

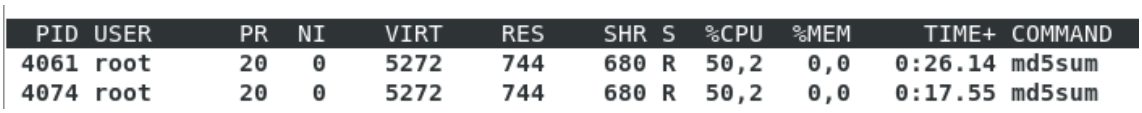
6. Indicar a cada uno de los cgroups creados en el paso anterior el porcentaje máximo de CPU que cada uno puede utilizar. El valor de cpu.shares en cada cgroup es 1024. El cgroup cpualta recibirá el 70 % de CPU y cpubaja el 30 %.
   # echo 717 > /sys/fs/cgroup/cpu/cpualta/cpu.shares
   # echo 307 > /sys/fs/cgroup/cpu/cpubaja/cpu.shares

7. Iniciar dos sesiones por ssh a la VM.(Se necesitan dos terminales, por lo cual, también podría ser realizado con dos terminales en un entorno gráfico). Referenciaremos a una terminal como termalta y a la otra, termbaja.

8. Usando el comando taskset, que permite ligar un proceso a un core en particular, se iniciará el siguiente proceso en background. Uno en cada terminal. Observar el PID asignado al proceso que es el valor de la columna 2 de la salida del comando.
   # taskset -c 0 md5sum /dev/urandom &

   termalta: [1] 4061
   termbaja: [1] 4074

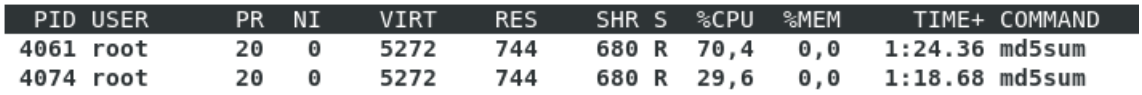
9. Observar el uso de la CPU por cada uno de los procesos generados (con el comando top en otra terminal). ¿Qué porcentaje de CPU obtiene cada uno aproximadamente?



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4061	root	20	0	5272	744	680	R	50,2	0,0	0:26.14	md5sum
4074	root	20	0	5272	744	680	R	50,2	0,0	0:17.55	md5sum

10. En cada una de las terminales agregar el proceso generado en el paso anterior a uno de los cgroup (termalta agregarla en el cgroup cpualta, termbaja en cpubaja. El process_pid es el que obtuvieron después de ejecutar el comando taskset)
 # echo "process_pid" > /sys/fs/cgroup/cpu/cpualta/cgroup.procs

11. Desde otra terminal observar cómo se comporta el uso de la CPU. ¿Qué porcentaje de CPU recibe cada uno de los procesos?



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4061	root	20	0	5272	744	680	R	70,4	0,0	1:24.36	md5sum
4074	root	20	0	5272	744	680	R	29,6	0,0	1:18.68	md5sum

12. En termalta, eliminar el job creado (con el comando jobs ven los trabajos, con kill %1 lo eliminan. No se olviden del %.). ¿Qué sucede con el uso de la CPU?

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4074	root	20	0	5272	744	680	R	100,0	0,0	1:35.91	md5sum

Es del 100% porque el proceso tiene todo ese núcleo disponible para el (no hay ningún otro proceso).

13. Finalizar el otro proceso md5sum.

14. En este paso se agregarán a los cgroups creados los PIDs de las terminales (Importante: si se tienen que agregar los PID desde afuera de la terminal ejecute el comando echo $$ dentro de la terminal para conocer el PID a agregar. Se debe agregar el PID del shell ejecutando en la terminal).
 # echo $$ > /sys/fs/cgroup/cpu/cpualta/cgroup.procs (termalta)
 # echo $$ > /sys/fs/cgroup/cpu/cpubaja/cgroup.procs (termbaja)

15. Ejecutar nuevamente el comando taskset -c 0 md5sum /dev/urandom & en cada una de las terminales. ¿Qué sucede con el uso de la CPU? ¿Por qué?

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 4713 root      20   0    5272    728    668 R  69,8   0,0   1:35.11 md5sum
 4720 root      20   0    5272    744    680 R  30,2   0,0   0:38.35 md5sum
```

Nuevamente lo mismo, el kernel limita el uso de CPU de las terminales y sus subprocesos (los iniciados con el taskset), haciendo que estos usen el %70 en el caso de termalta y el %30 en el caso de termbaja.

16. Si en termbaja ejecuta el comando: taskset -c 0md5sum /dev/urandom & (deben quedar 3 comandos md5 ejecutando a la vez, 2 en el termbaja). ¿Qué sucede con el uso de la CPU? ¿Por qué?

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 4713 root      20   0    5272    728    668 R  70,3   0,0   3:51.75 md5sum
 5086 root      20   0    5272    744    680 R  15,0   0,0   0:01.23 md5sum
 4720 root      20   0    5272    744    680 R  14,7   0,0   1:35.60 md5sum
```

Entre los dos procesos de termbaja usan ambos 30% de CPU.

Namespaces

1. Explique el concepto de namespaces.

Un namespace envuelve un recurso global del sistema en una abstracción que hace que parezca, para los procesos dentro del espacio de nombres, que tienen su propia instancia aislada del recurso global.

- Limitan lo que un proceso puede ver (y en consecuencia lo que puede usar)
- Modificaciones a un recurso quedan contenidas dentro del “namespace”. Los cambios en el recurso global son visibles para otros procesos que son miembros del mismo namespace, pero son invisibles para otros procesos.
- Un proceso solo puede estar en un namespace de un tipo a la vez.

- Un namespace es automáticamente eliminado cuando el último proceso en el finaliza o lo abandona.
- Un proceso puede utilizar ninguno/algunos/todos de los namespace de sus padre.
- Se utilizan para implementar contenedores.

2. ¿Cuáles son los posibles namespaces disponibles?

Namespace	***Flag***	***Isolates***
Cgroup	CLONE_NEWCGROUP	Cgroup root directory
IPC	CLONE_NEWIPC	System V IPC, POSIX message queues
Network	CLONE_NEWNET	Network devices, stacks, ports, etc.
Mount	CLONE_NEWNS	Mount points

PID	CLONE_NEWPID	Process IDs
Time	CLONE_NEWTIME	Boot and monotonic clocks
User	CLONE_NEWUSER	User and group IDs
UTS	CLONE_NEWUTS	Hostname and NIS domain name

3. ¿Cuáles son los namespaces de tipo Net, IPC y UTS una vez que inicie el sistema (los que se iniciaron la ejecutar la VM de la cátedra)?

```
root@so2022:~# lsns --type "net"
        NS TYPE NPROCS   PID USER     NETNSID NSFS COMMAND
4026531992 net     168     1 root  unassigned      /sbin/init
4026532277 net       1  1052 rtkit unassigned      /usr/lib/rtkit/rtkit-daemon
root@so2022:~# lsns --type "ipc"
        NS TYPE NPROCS PID USER COMMAND
4026531839 ipc     169   1 root /sbin/init
root@so2022:~# lsns --type "uts"
        NS TYPE NPROCS PID USER COMMAND
4026531838 uts _   169   1 root /sbin/init
```

4. ¿Cuáles son los namespaces del proceso cron? Compare los namespaces net, ipc y uts con los del punto anterior, ¿son iguales o diferentes?

```
root@so2022:~# pgrep cron
539
root@so2022:~# lsns --task 539
        NS TYPE   NPROCS PID USER COMMAND
4026531835 cgroup    168   1 root /sbin/init
4026531836 pid       168   1 root /sbin/init
4026531837 user      168   1 root /sbin/init
4026531838 uts       168   1 root /sbin/init
4026531839 ipc       168   1 root /sbin/init
4026531840 mnt       160   1 root /sbin/init
4026531992 net _     167   1 root /sbin/init
```

Son iguales

5. Usando el comando unshare crear un nuevo namespace de tipo UTS.

   a) unshare –uts sh (son dos (- -) guiones juntos antes de uts)

   b) ¿Cuál es el nombre del host en el nuevo namespace? (comando hostname)

      so2022

   c) Ejecutar el comando lsns. ¿Qué puede ver con respecto a los namespace?.

```
        NS TYPE   NPROCS   PID USER   COMMAND
4026531835 cgroup    177     1 root   /sbin/init
4026531836 pid       177     1 root   /sbin/init
4026531837 user      177     1 root   /sbin/init
4026531838 uts       175     1 root   /sbin/init
4026531839 ipc       177     1 root   /sbin/init
4026531840 mnt       169     1 root   /sbin/init
4026531860 mnt         1    30 root   kdevtmpfs
4026531992 net       176     1 root   /sbin/init
4026532154 mnt         1   319 root   /lib/systemd/systemd-udevd
4026532210 mnt         1   541 root   /usr/sbin/NetworkManager --no-daemon
4026532211 mnt         1   551 root   /usr/sbin/ModemManager --filter-policy=str
4026532221 uts         2  8161 root   sh
4026532277 net         1  1052 rtkit  /usr/lib/rtkit/rtkit-daemon
4026532330 mnt         1  1052 rtkit  /usr/lib/rtkit/rtkit-daemon
4026532386 mnt         1  1063 root   /usr/lib/upower/upowerd
4026532388 mnt         1  1170 colord /usr/lib/colord/colord
4026532389 mnt         1  1633 root   /usr/lib/fwupd/fwupd
```

      Hay un nuevo namespace uts para sh.

   d) Modificar el nombre del host en el nuevo hostname.

```
# hostname taylorswift
# hostname
taylorswift
```

e) Abrir otra sesión, ¿cuál es el nombre del host anfitrión?

```
root@so2022:~# hostname
so2022
```

Sigue siendo so2022 a pesar de que se cambió en el otro.

f) Salir del namespace (exit). ¿Qué sucedió con el nombre del host anfitrión?

Sigue siendo so2022, no se modificó.

6. Usando el comando unshare crear un nuevo namespace de tipo Net.
   a) unshare –pid sh

   b) ¿Cuál es el PID del proceso sh en el namespace? ¿Y en el host anfitrión?

```
# ps -C sh
  PID TTY          TIME CMD
 8829 pts/2    00:00:00 sh
```

```
root@so2022:~# ps -C sh
  PID TTY          TIME CMD
 8829 pts/2    00:00:00 sh
```

En ambos es el mismo. Esto es así por lo que se explica en c).

   c) Ayuda: los PIDs son iguales. Esto se debe a que en el nuevo namespace se sigue viendo el comando ps sigue viendo el /proc del host anfitrión. Para evitar esto (y lograr un comportamiento como los contenedores), ejecutar: *unshare --pid --fork --mount-proc*

   d) En el nuevo namespace ejecutar ps -ef. ¿Qué sucede ahora?

```
UID          PID  PPID  C STIME TTY          TIME CMD
root           1     0  0 15:38 pts/2    00:00:00 sh
root           3     1  0 15:38 pts/2    00:00:00 ps -ef
```

Ahora si son distintos ya que no se ve la carpeta /proc/ del host anfitrión.

e) Salir del namespace

Fuentes

https://www.geeksforgeeks.org/difference-between-virtualization-and-emulation/

https://aws.amazon.com/es/what-is/virtualization/#:~:text=La%20virtualizaci%C3%B3n%20es%20una%20tecnolog%C3%ADa,en%20una%20%C3%BAnica%20m%C3%A1quina%20f%C3%ADsica.

https://mohitdtumce.medium.com/understanding-full-virtualization-paravirtualization-and-hardware-assist-730d3c9aa04c

https://www.baeldung.com/cs/virtualization-techniques-compared

https://www.ibm.com/mx-es/topics/containers

https://man7.org/linux/man-pages/man7/namespaces.7.html