Sistemas operativos avanzados

Planificación del procesador

3ª parte: soporte para los contenedores

Abstracciones "tradicionales" de los SS.OO.

- 2 abstracciones básicas: "máquina" y "proceso"
- SO proporciona una "máquina" extendida
  - Con mayor nivel de abstracción que la máquina real:
    - Puertos y sockets en vez de tarjetas de red
    - Ficheros en vez de discos,…
- SO provee la abstracción "proceso"
  - Programa en ejecución que cree tener su propia máquina
  - SO reparte equitativamente UCP y memoria entre procesos
    - Pudiendo limitar su uso de recursos (UNIX *setrlimit*)
  - Procesos comparten/(compiten por) recursos de la "máquina"
    - Ficheros, puertos,…
    - Visión común de los recursos compartidos
      - El fichero */dir1/dir2/f* es el mismo para todos los procesos
      - Dos servidores web no pueden usar el puerto 80

Abstracciones requeridas por los contenedores

- Procesos no son siempre entidades individuales independientes
  - Ciertos procesos están relacionados entre sí
    - P.e. procesos que forman parte de una aplicación
  - Se puede necesitar
    - Asignar/limitar el uso de recursos **a todo el grupo de procesos**
    - Proporcionar al grupo su propia visión de la máquina
      - Su propio árbol de ficheros, sus propios puertos,…
- SS.OO. no ofrecían esta abstracción
  - UNIX soporte de grupos de procesos sólo para casos puntuales
    - P.e. procesos en la misma sesión o en el mismo *pipeline*
- Se requiere abstracción que permita definir grupos de procesos
  - Asignando/limitando recursos al grupo
  - Proporcionando al grupo su propia máquina extendida

Soporte de contenedores en Linux

- Basado en dos componentes básicos: ***cgroups*** y ***namespaces***
- ***Cgroups***
  - Definición de grupos y asignación de recursos a los mismos
  - Actualmente en transición de v1 a v2
    - Presentación basada en v1
- ***Namespaces***
  - Visión propia de cierto tipo de recurso para un proceso/grupo
- Diseñados como componentes independientes y ortogonales
  - Su uso combinado para crear contenedores es solo un caso
  - La imaginación es el límite…
- Algunas referencias:
  - *https://lwn.net/Articles/531114/*
  - *https://lwn.net/Articles/604609/*
  - *http://www.netdevconf.org/1.1/proceedings/slides/rosen-namespaces-cgroups-lxc.pdf*

Namespaces

- 6 implementados que pueden usarse de forma independiente:
  - Espacios de nombres para operaciones de montaje
  - Espacios de nombres para PIDs
  - Espacios de nombres para la red
  - Espacios de nombres para IPCs
  - Espacios de nombres para nombre de máquinas (UTS)
  - Espacios de nombres para usuarios
- SO Plan 9 de laboratorios Bell precursor de la idea
- Crear espacios de nombres requiere ser administrador
  - Excepto para espacios de nombres de usuarios
    - Un usuario normal puede crear un espacio de nombres de usuario
      - Donde puede ser administrador

Namespaces: API

- Afecta a tres llamadas (sólo una nueva: *setns*)
- ***clone*** (ya conocida): crea un nuevo proceso hijo
  - En un nuevo espacio de nombres si se especifica *flag*:
    - CLONE_NEWNS CLONE_NEWUTS CLONE_NEWIPC CLONE_NEWPID CLONE_NEWNET CLONE_NEWUSER
- ***unshare***: crea nuevo espacio de nombres acorde a mismo *flag*:
  - Proceso que realiza llamada se hace miembro del mismo
- ***setns***: proceso se hace miembro de espacio nombres existente
  - Los espacios de nombres son anónimos
  - Se identifican por el fichero correspondiente en */proc*
- Jerarquía de espacios de nombres anidados
  - Proceso en nuevo espacio de nombres crea otro espacio
- Mandatos: *unshare*, *nsenter*,…

Ficheros de espacios de nombres

```
$ ls -l /proc/$$/ns
total 0
lrwxrwxrwx 1 ssoo ssoo 0 Mar 28 19:57 ipc -> ipc:[4026531839]
lrwxrwxrwx 1 ssoo ssoo 0 Mar 28 19:57 mnt -> mnt:[4026531840]
lrwxrwxrwx 1 ssoo ssoo 0 Mar 28 19:57 net -> net:[4026531956]
lrwxrwxrwx 1 ssoo ssoo 0 Mar 28 19:57 pid -> pid:[4026531836]
lrwxrwxrwx 1 ssoo ssoo 0 Mar 28 19:57 user -> user:[4026531837]
lrwxrwxrwx 1 ssoo ssoo 0 Mar 28 19:57 uts -> uts:[4026531838]
```

Breve reseña de cada espacio de nombres

- CLONE_NEWNS
  - Ops. montaje en nuevo espacio no afectan a espacio padre
  - Procesos en el e. de nombres tienen su propio árbol de ficheros
- CLONE_NEWUTS
  - Nombre de máquina y dominio específico para cada e. nombres
- CLONE_NEWIPC
  - Cada espacio de nombres gestiona IDs para IPCs de SystemV
- CLONE_NEWPID
  - Cada espacio de nombres gestiona su propio rango de PIDs
  - Primer proceso en espacio de nombres recoge huérfanos

Breve reseña de cada espacio de nombres

- CLONE_NEWNET
  - Virtualización de la pila de red
  - Cada espacio de nombres gestiona sus propios:
    - IPs, puertos, sockets, tablas de encaminamiento,…
  - Nuevo espacio de nombres sólo incluye *loopback*
  - Cada interfaz de red asignado a sólo un espacio de nombres
    - Pero se puede mover entre espacios de nombres
  - Mandato: *ip netns*
- CLONE_NEWUSER
  - Pueden establecerse correspondencias entre:
    - UIDs en nuevo espacio y UIDs en el espacio padre
    - GIDs en nuevo espacio y GIDs en el espacio padre
  - UID 0 en nuevo espacio puede asociarse a UID normal en padre

cgroups

- No requiere nuevas llamadas
- Funcionalidad accesible:
  - Montando un sistema ficheros de tipo *cgroup* en un directorio
    - En distribuciones basadas en *systemd* se monta en directorio:
      - */sys/fs/cgroup*
  - Especificando como opción el tipo de recurso a controlar
  - Y creando directorios y leyendo/escribiendo ficheros en ese dir.
  - Cada directorio de recurso incluye (entre otros)
    - Fichero *tasks* que contiene los procesos asociados a ese grupo
    - Un fichero por cada opción configurable (p.e. *cpu.shares*)
- Gestiona recursos relacionados con UCP, memoria y E/S
  - Control de recursos de red por grupos mediante *iptables* y *tc*

cgroups: Ejemplo

https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt

```
$ mount -t tmpfs cgroup_root /sys/fs/cgroup
$ mkdir /sys/fs/cgroup/cpu
$ mount -t cgroup -ocpu none /sys/fs/cgroup/cpu
$ cd /sys/fs/cgroup/cpu
$ mkdir multimedia
# create "multimedia" group of tasks
$ mkdir browser
# create "browser" group of tasks
#Configure the multimedia group to receive twice the CPU bandwidth that of browser group
$ echo 2048 > multimedia/cpu.shares
$ echo 1024 > browser/cpu.shares
$ firefox & # Launch firefox and move it to "browser" group
$ echo $!  > browser/tasks  # <firefox_pid>
$ gmplayer & # Launch gmplayer and move it to “multimedia" group
$ echo $! > multimedia/tasks # <movie_player_pid>
```

cgroups: Otro ejemplo

```
$ mkdir /sys/fs/cgroup/cpu
$ mount -t cgroup -omemory none /sys/fs/cgroup/memory
$ mkdir /sys/fs/cgroup/memory/group0
$ echo $$ > /sys/fs/cgroup/memory/group0/tasks
$ echo 40M > /sys/fs/cgroup/memory/group0/memory.limit_in_bytes
```

Breve reseña de cada cgroup

- *blkio*: límites en E/S a dispositivos de bloques
- *cpuacct*: contabilidad uso de UCP del grupo
- *cpu*: información para planificación de procesos del grupo
- *cpuset*: asignación de UCPs y nodos a procesos del grupo
- *devices*: limita accesos a dispositivos a procesos del grupo
- *freezer*: permite suspender procesos del grupo
- *hugetlb*: contabilidad uso páginas grandes por procesos del grupo
- *memory*: control del uso de memoria por procesos del grupo
- *net_cls*:permite identificar paquetes asociados a procesos del grupo
- *net_prio*: permite priorizar paquetes asociados a procesos del grupo
- *perf_event*: datos de rendimiento de procesos del grupo
- *pids*: limita el nº de PIDs usados por los procesos del grupo