Práctica 6

Rendimiento y Afinamiento del Sistema	
OBJETIVOS	Saber usar las herramientas de monitorización del funcionamiento del sistema para ser capaces de identificar situaciones que afecten al rendimiento general. Saber usar las herramientas que modifican el funcionamiento de algún componente del sistema con el fin de mejorar el rendimiento general.
TAREAS	Usar las órdenes **uptime, ps, vmstat y top** con el objeto de monitorizar diferentes aspectos básicos del sistema (carga de trabajo, entrada/salida, memoria virtual, etc...). Cambiar prioridades y momentos de ejecución de procesos con **nice**, **renice**, **at** y **batch**. Realizar actividades de afinamiento del sistema que mejoren su rendimiento: equilibrar la carga de trabajo, crear nuevas áreas de intercambio (*swap*), crear sistemas de archivos en RAM… Usar la funcionalidad de *Cgroups* para controlar la asignación de recursos a aplicaciones de usuario y servicios del sistema mediante las herramientas que proporciona **systemd**.
DOCUMENTACIÓN	Diapositivas del Tema 6 sobre «Monitorización» y "Control groups (Cgroups)"
TIEMPO ESTIMADO	3 horas en laboratorio

Tareas propuestas

Se propone realizar las siguientes actividades:

1. Monitorizar la carga de trabajo con las órdenes `uptime` y `ps`

La orden **`uptime`** nos proporciona información general acerca de la carga de trabajo del sistema. Se trata de entender el significado de los datos que nos proporciona. Se propone como actividad:

- Obtener el tiempo que lleva funcionando el sistema, el número de usuarios que utilizan el sistema y la carga media.

Aunque la utilidad **`ps`** ya se ha utilizado, ahora en este punto se trata de utilizarla para conocer información relativa a los procesos y que afecta a la carga de trabajo del sistema. Concretamente se trata obtener información de los procesos sobre el tiempo consumido de CPU, memoria ocupada y estado. Como actividades de monitorización usando la orden **`ps`** se propone:

- Obtener un listado de los procesos ordenados de mayor a menor por tiempo de CPU consumido.

- Obtener un listado de los procesos ordenados de mayor a menor por espacio de memoria consumido.

2. Monitorizar los recursos básicos del sistema (CPU, memoria, swap y entrada/salida) con la orden `vmstat`

La orden **`vmstat`** nos proporciona información que nos permite conocer el estado de uso de los recursos básicos del sistema y las tareas que se han creado desde que el sistema arrancó mediante las llamadas al sistema **`fork`**, **`vfork`** y **`clone`**. Esta orden se puede invocar para que periódicamente nos devuelva un informe dicho estado de uso. Como actividad de monitorización usando esta orden se propone:

- Obtener cada hora un informe del sistema que incluya la memoria en uso y la no utilizada. Cada informe debe almacenarse añadiéndose al contenido de un fichero que recibirá el nombre **`/root/InfoSys`**.

- Obtener un informe de los discos del sistema cada hora. Cada informe debe almacenarse añadiéndose al contenido de un fichero de nombre /**`root/InfoDisks`**.

3. Monitorizar el sistema y los procesos de forma dinámica con la orden `top`

La utilidad **`top`** nos informa del estado del sistema y el uso de los recursos por parte de los procesos que se ejecutan en el sistema. Se trata de una utilidad interactiva, es decir cuándo se invoca nos proporciona un conjunto de órdenes que nos permite obtener distinto tipo de información y en distintos formatos. Esta orden se puede invocar para que nos proporcione información de todo el sistema o de un conjunto de procesos específicos de forma periódica y en un número de iteraciones que se especifica cuando se invoca la orden. La orden se usará de distintas formas con el objeto de obtener distinta información sobre el sistema y los procesos que se ejecutan. Concretamente se propone utilizar esta orden para:

- Obtener información general del sistema y de los procesos.

- Obtener 4 informes sobre el rendimiento general del sistema y de los procesos. Entre cada informe deben transcurrir 10 segundos.

- Monitorizar los 10 procesos que más tiempo llevan ejecutándose en el sistema.

- Monitorizar los procesos de un usuario mostrando PID, uso de CPU, uso de memoria y tiempo que llevan ejecutándose.

4. Afinamiento del sistema a través de mejoras en su carga de trabajo: configuración en el arranque de aquellos servicios necesarios mediante la orden `systemctl`

Una forma de mejorar el rendimiento del sistema es configurarlo para que sólo ejecute aquellos servicios que sean necesarios. La orden **`systemctl`** nos permite arrancar, detener u obtener información de estado de servicios y, además, nos permite configurar el sistema para que durante el arranque sólo se inicien aquellos servicios que establezcamos. Se propone realizar las siguientes tareas:

- Obtener los servicios que se inician automáticamente durante el arranque del sistema.

- Para algunos de ellos obtener sus estados, recargar nuevas configuraciones sin reiniciarlos, pararlos y volverlos a ejecutar.

- Averiguar si hay servicios que se inician automáticamente con el arranque del sistema y, si es así, entonces configurar el sistema para que no se inicien con el arranque.

5. Afinamiento del sistema a través de mejoras en su carga de trabajo: redefinir las prioridades de ejecución de un proceso usando las órdenes `nice` *y* `renice`

Otra forma de mejorar el rendimiento del sistema es configurar las prioridades de ejecución de los procesos que se ejecutan en el sistema, de forma que se prioricen aquellos que nos interese que se ejecuten antes en detrimento de otros que se consideren no prioritarios.

La orden **`nice`** nos permite modificar las prioridades de ejecución de los procesos. Se propone realizar las siguientes tareas:

- Ejecutar una tarea con una prioridad dada.

- Alterar las prioridades de un proceso o conjunto de procesos que se están ejecutando.

6. Afinamiento del sistema a través de mejoras en su carga de trabajo: planificar la ejecución de tareas en momentos en los que el sistema tiene poca carga de trabajo usando las órdenes `at` *y* `batch`

Otra forma de mejorar el rendimiento del sistema es planificar de ejecución de aquellas tareas que no tienen por qué ejecutarse de forma inmediata, de manera que se ejecuten en momentos en los que el sistema posea una carga de trabajo menor. La orden **`at`** nos permite planificar la ejecución de una tarea en un momento dado (fecha y hora). Nótese la diferencia entre lo que hace esta orden y lo que hace **`crontab`**.

Se propone realizar las siguientes tareas:

- Planificar la ejecución de una orden en una fecha y hora dada.

- Ver el estado de de la cola de tareas **`at`**.

- Eliminar una tarea de la cola.

Podemos aprovechar con comodidad la orden **`at`** si conocemos de antemano las cargas de trabajo del sistema y por tanto podemos anticipar cuándo es el mejor momento para ejecutar tareas de bajar prioridad. Cuando la carga del sistema es menos previsible, podemos utilizar la orden **`batch`**. Esta orden ejecuta las tareas de su cola sólo cuando la carga de trabajo está por debajo de un determinado valor (configurable). Para practicar con **`batch`** se propone:

- Configurar el umbral de carga de trabajo para la ejecución de las tareas de la cola **`batch.`**

- Planificar la ejecución de una tarea.

- Ver el estado de de la cola de tareas **`batch`**.

- Eliminar una tarea de la cola.

7. Afinamiento del sistema controlando los recursos que las aplicaciones de usuario y los servicios del sistema pueden utilizar: Grupos de Control (Cgroups)

Los Grupos de Control (*Cgroups*) son una funcionalidad del núcleo Linux que permite asignar recursos (CPU, memoria, ancho de banda de E/S, ancho de banda de red, etc...) a grupos de procesos. Estos procesos pueden ser programas que ejecutan los usuarios o procesos pertenecientes a un servicio del sistema. Mediante *Cgroups*, los recursos se asignan a un proceso y a sus procesos descendientes. Esta funcionalidad permite un control fino de la cantidad de recursos que un programa o servicio utiliza, así como priorizar uso de los recursos entre los programas o servicios. Nótese que con el uso de *Cgroups* no solo se posibilita una asignación de recursos para que las aplicaciones y servicios se ejecuten adecuadamente, también se mejora la seguridad del sistema, ya que evita que un determinado servicio o aplicación colapse el sistema debido a un consumo desmesurado de recursos.

Cgroups se puede manejar de dos formas diferentes. La primera consiste en utilizar las herramientas que proporciona la librería **`libcgroup`**. La segunda manera consiste en utilizar órdenes que proporciona el servicio **`systemd`**. En general, la librería **`libcgroup`** se considera obsoleta, pero se sigue manteniendo por

motivos de compatibilidad en las distribuciones actuales de Red Hat y en las distribuciones derivadas. Por este motivo, se recomienda el manejo de *Cgroups* mediante **systemd**. Teniendo presente esta recomendación, en esta actividad práctica se utilizarán las órdenes proporcionadas por **systemd**.

La asignación de recursos, *Cgroups*, a un proceso y sus descendientes puede ser transitoria o permanente. Un *Cgroup* transitorio se define en la orden de ejecución del programa y permanece mientras el programa se ejecuta. Una vez el programa finaliza su ejecución, el *Cgroup* desaparece. La orden que proporciona **systemd** para crear un *Cgroup* transitorio es **systemd-run**. Una sintaxis muy frecuente de esta orden es:

```
# systemd-run --unit=name --scope --slice=slice_name command
```

- Crear un *Cgroup* transitorio para un programa. Por ejemplo, crear un *Cgroup* para una sesión de la orden **top**.

- Obtener información acerca de la nueva unidad creada.

- Limitar el uso de un recurso, por ejemplo, la cantidad de memoria, durante la ejecución de la unidad creada anteriormente.

Normalmente, los *Cgroups* se utilizan en el contexto de los servicios del sistema. Su uso en los servicios del sistema permite una asignación de recursos para que éstos se puedan ejecutar adecuadamente y no acaparen recursos. Seguidamente, se planten actividades de manejo básico de servicios con *Cgroups*.

- Lanzar un servicio **httpd** y crearle un *Cgroup* transitorio. Concretamente, se propone lanzar el servicio **httpd** limitando el porcentaje de uso que puede utilizar este servicio en una CPU y la cantidad máxima de memoria que pueda consumir. Estos límites máximos son 25% para la CPU a 300 Mbytes para la memoria.

- Crear un *Cgroup* permanente para un servicio. Se propone crear un *Cgroup* permanente para el servicio **httpd** con las mismas limitaciones de recursos especificadas en la actividad anterior.

- Crear un *Cgroup* permanente en un servicio que se lanza automáticamente con el arranque del sistema. Se propone crear un *Cgroup* permanente para el servicio **httpd** con las mismas limitaciones de recursos especificadas en la actividad anterior.

Seguidamente se aborda cómo obtener información de los *Cgroups* existentes en el sistema. En primer lugar, hay que recordar que los *Cgroups* se asocian a unidades gestionadas por **systemd** de tipo *Scope* o de tipo Service y que una o varias unidades de este tipo pueden dar lugar a unidades de tipo Slice. Por ello, es útil obtener las unidades existentes en el sistema que sean de estos tipos.

- Obtener información sobre las unidades existentes en el sistema.

- Obtener información de los *Cgroups* existentes en el sistema.

- Obtener el consumo de recursos de las distintas unidades activas del sistema.

8. Afinamiento del sistema a través de la mejora en la gestión de la memoria del sistema: área de swap, órdenes `fdisk`, `mkswap` y `swapon`

La gestión de la memoria del sistema influye de manera muy importante en el rendimiento del sistema. La mayoría de las actuaciones de afinamiento del comportamiento del sistema de gestión de memoria se realizan modificando parámetros de configuración que forman parte del núcleo. Por tanto, para que tengan efecto, se debe compilar un nuevo núcleo que posea estos nuevos valores. En nuestro, caso vamos a realizar una tarea relacionada con la gestión de memoria que no requiere compilar un nuevo núcleo. Concretamente, se propone crear y activar una nueva área de *swap* del sistema. Para ello se deben utilizar las órdenes `fdisk`, `mkswap` y `swapon`.

Esta clase de tarea se debe acometer cuando se ha detectado que la configuración actual del área de *swap* no es la apropiada como, por ejemplo, cuando es demasiado pequeña y por ello el sistema deja de ejecutar nuevos programas debido a que no tiene memoria. También la usaríamos cuando se constata que el área de *swap* es demasiado grande y se está desaprovechando espacio de disco. Se propone:

- Crear una nueva partición en el sistema para albergar una nueva área de *swap*.
- Crear, en la nueva partición, un área de *swap*.
- Configurar el sistema para que utilice la nueva área de *swap*.

9. Afinamiento del tiempo de acceso a ficheros críticos del sistema

En algunas ocasiones existen datos que deben ser accedidos lo más rápidamente posible. Una forma de garantizar el acceso rápido a estos datos es ubicarlos en un sistema de archivos que resida en la memoria principal del sistema. En esta tarea se propone crear un sistema de archivos en memoria, para ello se deberá seguir el siguiente procedimiento:

1. Crear el directorio que se utilizará como punto de montaje cuando se monte el sistema de archivos a crear.

2. Asignar los permisos de acceso adecuados al directorio creado.

3. Crear el sistema de archivos utilizando la orden `mount` y las opciones `-t` y `-o` de la orden.