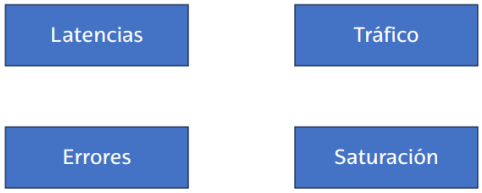
MONITORIZACIÓN

INTRODUCCIÓN

Para que una aplicación funcione bien hay que mirar los siguientes aspectos:

- Latencia:
  - El tiempo necesario para completar una petición.
  - Importante distinguir entre peticiones satisfactorias y fallidas.
- Tráfico:
  - Demanda de un sistema.
  - Suelen ser específicas de un sistema.
  - Ejemplos:
    - Peticiones HTTP / segundo.
    - Transacciones / segundo
- Errores:
  - Resultado de peticiones fallidas.
  - Pueden ser explícitos o implícitos
  - Ejemplo: Al solicitar una web a un servidor:
    - Error explícito: error 404.
    - Error implícito: recuperar una web diferente a la esperada.
- Saturación:
  - Medida de la capacidad de un recurso en uso.
  - Importante observar los recursos más limitados.
  - Latencia puede ser un indicador de saturación



Latencias	Monitorizar logs de aplicación.
Tráfico	Monitorizar logs de aplicación y red.
Errores	Monitorizar logs de aplicación y del sistema.
Saturación	Monitorizar hardware (CPU, memoria, ...) y red.

MONITORIZACIÓN DE CPU

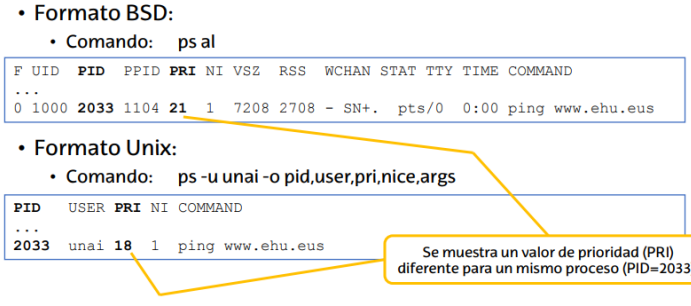
- Comando top → Uso de recursos del sistema en tiempo real
- Comando ps → Listado de procesos y su uso de recursos
- Comando pstree → Árbol de procesos del sistema

GESTIÓN DE RECURSOS

GESTIÓN DE PROCESOS

Prioridades de los procesos

- El planificador del SO asigna intervalos de tiempo a los procesos según su prioridad.
- Esto se controla según 2 valores (aparecen en top por ejemplo):
  - Prioridad (PR): puede tomar valores en el rango -100 a 39.
  - Valor "nice" (NI): puede tomar valores en el rango -20 a 19.
- Para ambos, cuanto **más negativo el valor, mayor prioridad**.
- En Linux, los procesos se consideran de 2 tipos:
  - **Procesos normales (la mayoría de los lanzados por usuarios).**
  - **Procesos de tiempo real (generalmente, los esenciales para el SO)**
- Prioridades de los procesos normales
  - Se calcula: PR = 20 + NI
    - P.e. si el valor NI de un proceso es -20, su prioridad es 0
    - P.e. si el valor NI es 19, su prioridad es 39.
  - Procesos normales → Rango 0-39 de prioridades.
  - Por defecto, "nice" (NI) es 0.
    - Un usuario normal puede modificar el valor NI entre 0 y 19.
      - Puede reducir la prioridad sobre el resto de procesos del sistema
    - El usuario root puede modificar el valor NI -20 y 19.
  - **Comando nice**
    - Lanza un comando con un valor Nice concreto
    - Sintaxis: nice -n valor comando → Valor es relativo (define más o menos)
      - Ejemplo: nice -n 10 ./miScript
  - **Comando renice**
    - Cambia el valor Nice de un proceso (o grupo) en ejecución
    - Un usuario normal (no root) sólo puede incrementar el valor (irreversible)
    - Sintaxis: renice -n valor -p PID [-g grupo] → Valor es absoluto
      - Ejemplo: renice -n 15 -p 7552
- Prioridades de los procesos en tiempo real
  - Se calcula: PR = - 1 - prioridad_tiempo_real.
    - prioridad_tiempo_real toma valores entre 1 y 99.
      - P.e. si prioridad_tiempo_real es 50, PR vale -51
  - El valor "nice" no se tiene cuenta
  - En el comando top, si PR=-100, se muestra como 'rt' (real time).
  - **Comando chrt** → Lanza un proceso con una prioridad de tiempo real
    - Sintaxis: chrt --rr
      - Ejemplo: chrt --rr 20 ./miPrograma
      - Lanzaría ./miPrograma con PR=-21
- **Comando ps**
  - Puede mostrar datos en diferentes formatos
  - Cada formato puede mostrar una misma prioridad con diferentes valores



- **Comando kill**
  - Envía señales a procesos
  - Sintaxis: kill PID
  - Opciones:
    - -l Mostrar las señales disponibles
    - señal Mandar una señal al proceso
  - 3 formas de in dicar una señal:
    - Número -19
    - Prefijo SIG -SIGSTOP
    - Sin prefijo SIG -STOP
  - Señales:
    - -STOP Parar el proceso
    - -CONT Reanudar el proceso (parado con STOP)
    - -KILL Matar el proceso
- **Comando ulimit**
  - Limitar el uso de recursos
  - Los límites sirven para la Shell en uso
  - Sintaxis: ulimit - [límite]
  - Opciones:
    - -a Lista los límites establecidos
    - -f Máximo número de ficheros creados por la Shell
    - -m Máxima memoria disponible
    - -t Máximo tiempo de CPU (segundos)

```
unai@unai-server:~$ ulimit -a
core file size          (blocks, -c) 0
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 31543
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
```

- **Fichero /etc/security/limits.conf**
  - Permite hacer una configuración permanente de límites
  - Cada línea tiene el siguiente formato: usuario/grupo tipo-de-límite ítem valor
    - usuario/grupo — Nombre del usuario o grupo (comienza con @)
    - tipo-de-limite — soft/hard
    - item — Puede ser: cpu, nproc, maxlogins, fsize, …
    - valor — Valor para el ítem definido

```
@student hard nproc 20
@faculty soft nproc 20
@faculty hard nproc 50
ftp      hard nproc 0
```

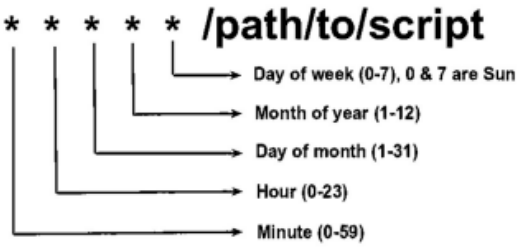
- Comando cpulimit
  - Permite limitar el % de uso constante de CPU de un proceso
  - ulimit y limits.conf sólo permiten limitar el tiempo total de uso CPU
  - nice y renice permiten reducir la prioridad pero no fijar un umbral
  - Uso → cpulimit --pid PID --limit
    - Donde <límite> es el límite de % CPU máximo que queremos permitir

PLANIFICACIÓN DE TAREAS

Se pueden programar tareas para que se ejecuten periódicamente (cron) o una única vez (atd). Estos leen periódicamente (por defecto, cada minuto) sus ficheros de configuración para ejecutar tareas programadas. Algunas tareas programables:

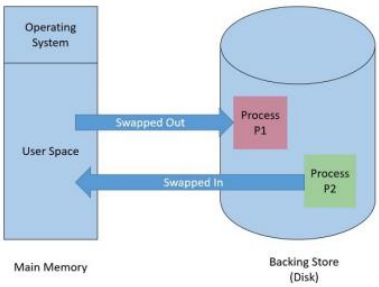
- Rotación de logs
- Borrar la carpeta /tmp
- Copias de seguridad
- Actualizar una BBDD

Comando crontab → https://crontab.guru/



- Una línea por tarea programada
- Sintaxis: crontab
- Opciones:
  - -l Mostrar las tareas programadas
  - -e Editar las tareas programadas
  - -r Elimina las tareas programadas
- Ejemplos:
  - 6 17 * * * /scripts/copia.sh Ejecuta copia.sh los sabados a las 17:00
  - * * * * * /scripts/miScript.sh Ejecuta miScript.sh cada minuto

GESTIÓN DE MEMORIA



La mayoría de sistemas operativos modernos utilizan memoria virtual y utilizan un espacio de disco como extensión de la memoria principal (espacio de intercambio o Swap). La memoria virtual se organiza en páginas que se intercambian entre memoria y disco.

- Un uso excesivo de Swap puede degradar el rendimiento
- Valores referencia de latencias en una jerarquía de memoria:

Tipo de memoria	Latencia
Caché L1 en CPU	1 ns
Caché L2 en CPU	4 ns
RAM	100 ns
SSD	16.000 ns
HDD	2.000.000 ns

Monitorizar la memoria

- Comando top → Utilizar Shitf+m (o Shift+f para entrar en el editor de orden) para ordenar por consumo descendente de memoria
- Comando vmstat → Campos relativos a la memoria:

```
procs  -----------memoria---------- ---swap-- -----io---- -sistema-- ------cpu-----
r  b   swpd   libre  búfer   caché   si   so    bi    bo   in   cs us sy  id wa st
1  0   8972 1097844 776348 5687216    0    0     0     3    1    2  0  0 100  0  0
0  0   8972 1097712 776348 5687216    0    0     0     0   42   48  0  0 100  0  0
0  0   8972 1097712 776348 5687216    0    0     0     0   39   48  0  0 100  0  0
```

swpd: Memoria swap en uso
libre: Memoria libre
buff: Memoria usada como buffer
cache: Memoria usada como cache

si: Memoria swap retirada de disco(Swap In)
so: Memoria swap llevada a disco (swap out)

GESTIÓN DE DISCOS Y FICHEROS

Monitorización:

- **Comando df**
  - Listado de sistemas de ficheros y espacio disponible
  - Sintaxis: df <opciones>
  - Ejemplo: df -h → Muestra los tamaños en kB, MB, … en lugar de en bytes
- **Comando du**
  - Tamaño de una rama del sistema de ficheros (p.e., de un directorio)
  - Sintaxis: du <opciones> directorio
  - Ejemplos: du -sh /home → Muestra el tamaño total del directorio /home

- **Comando lsof**
  - Muestra los ficheros en uso por los procesos del sistema (list open files)
  - Útil para resolver el **error "resource is busy"** al desmontar una partición.
- **Comando iostat**
  - Muestra estadísticas de uso y tasas de transferencia de los dispositivos de almacenamiento
  - Uso: iostat –p <disco>
    - Ejemplo: iostat –p /dev/sda

Device	tps	kB_read/s	kB_wrtn/s	kB_dscd/s	kB_read	kB_wrtn	kB_dscd
sda	2.26	26.52	258.30	127.34	438959	4275565	2107840
sda1	2.24	26.01	258.30	127.34	430614	4275564	2107840
sda14	0.00	0.02	0.00	0.00	272	0	0
sda15	0.01	0.43	0.00	0.00	7088	1	0

GESTIÓN DE RED

Ante un ataque por red o cualquier otro problema del tipo, se quita el cable de red en última instancia si no podemos solucionarlo, pero nunca el de alimentación.

Monitorización

- **Comando netstat**
  - Muestra información sobre las conexiones y rutas de red
  - Mostrar conexiones activas: **netstat -a | more**
  - Mostrar tabla de rutas: **netstat -r**
- **Comando nethogs**
  - Muestra conexiones y ratio de tráfico enviado/recibido
  - Requiere instalar el paquete sysstat
- **Comando tcpdump**
  - Es un analizador de tráfico para conexiones TCP/IP
  - Uso más común: captura de tráfico para posterior análisis.
  - Comenzar a capturar tráfico y guardar en un fichero
    - Sintaxis: tcpdump -i <interfaz> -Z <usuario> -w <ficheroCaptura>
    - Ejemplo: tcpdump -i ens4 -Z unai -w miCaptura
  - Las interfaces disponibles se pueden mostrar con: ip link
  - Visualizar un fichero de captura de tráfico:
    - Sintaxis: tcpdump -enr <ficheroCaptura>
    - Se puede añadir el parámetro -ttt para incluir la diferencia de tiempo entre cada paquete
- **Comando telnet**
  - Útil para comprobar si un servicio remoto está a la escucha.
  - Sintaxis: **telnet <IP> <puerto>**
- **Comando netcat**
  - Herramienta para leer de y escribir en conexiones de red.
  - Utilidad: Abrir una conexión a la escucha en un puerto.
    - Sintaxis: **nc -l <puerto>**
  - Utilidad: Conectarse a una IP/puerto y escribir en él.
    - Sintaxis: **nc <IP> <puerto>**

REGISTROS DEL SISTEMA (LOG)

LOGS

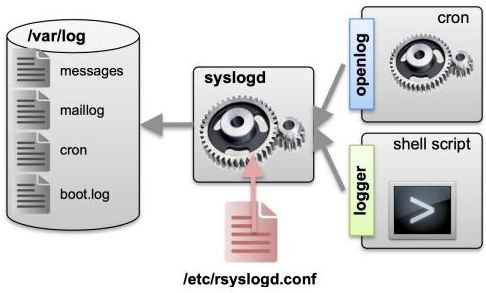
El kernel de Linux, los servicios y las aplicaciones generan eventos constantemente:

- Información sobre su estado
- Información sobre fallos/anomalías
- Errores de arranque
- Acceso a información (seguridad)

Una gestión adecuada de esta información es esencial para descubrir y solucionar problemas. Todos estos eventos suelen estar gestionados por un único servicio → En Unix/Linux es **syslog**

SYSLOG

Recolector de eventos empleado por el kernel, servicios y aplicaciones. Es flexible, seguro y fácil de usar. Está compuesto por los siguientes elementos:



- **syslogd**: Servicio del sistema → Recibe los mensajes del resto de servicios y aplicaciones y los añade al registro.
- **openlog**: Librerías para usar syslog desde una aplicación.
    - P.e., openlog (C/C++), sys::syslog(openlog(),syslog()) (Perl)
- **logger**: Comando del sistema para enviar mensajes a syslog.
- **rsyslogd.conf:** Fichero de configuración.
    - Listado de acciones a realizar en función de los mensajes recibidos
    - Tiene una línea por acción, con el formato:
        - entidad.nivel acción
    - **Entidad**: lista de valores definidos por el sistema → Kern, user, daemon (otro servicio), auth (login, su, ssh), mail, cron, …
    - **Nivel**: tipo de notificación → emerg, alert, crit, err, warning, notice, info, debug, * (todos los niveles)
    - **Acción**:
        - <nombre-de-fichero> Escribir el mensaje a ese fichero.
        - <nombre-dominio>/<IP> Enviar el mensaje al syslogd del nodo indicado.
        - <nombre-usuario> Enviar mensaje al usuario, si está conectado.
        - * Enviar mensaje a todo usuario conectado

```
# First some standard log files.  Log by facility.
#
auth,authpriv.* /var/log/auth.log
*.*;auth,authpriv.none -/var/log/syslog
#cron.* /var/log/cron.log
mail.* -/var/log/mail.log
#user.* -/var/log/user.log


# Logging for the mail system.
#
#mail.info -/var/log/mail.info
#mail.warn -/var/log/mail.warn
mail.err /var/log/mail.err
…
```

- Syslog escribe en ficheros en /var/log:
  - /syslog Eventos generales, ni críticos ni de depuración
  - /maillog Información de e-mails
  - /cron Registros del proceso cron
  - /boot.log Mensajes e información de inicio del sistema
- Hay ficheros en /var/log/ que no gestiona syslog
  - /wtmp Registra accesos de los usuarios y reinicios
    - Está en formato binario
    - Es utilizado por los comandos last y uptime
  - /lastlog Contiene el último acceso de cada usuario
  - /dmesg Eventos del inicio del sistema
    - Es utilizado por el kernel y proceso init

GESTIÓN DE LOGS

Cuanta más información de logs, mayor uso de disco. Los logs pueden llegar a consumir un espacio significativo y puede ser costoso buscar información/datos concretos entre miles de líneas.

Rotación de logs

Periódicamente cambiar el fichero donde se escriben los logs, cambiando a escribir en uno nuevo y borrando el más antiguo. Se puede hacer de manera manual con un script.

```
#!/bin/bash
cd /var/log/
mv messages.2 messages.3
mv messages.1 messages.2
mv messages messages.1
cat /dev/null > messages
chmod 600 messages

#Reiniciar syslog
service rsyslog restart
```

Se puede implementar la rotación con un servicio del sistema. Evita errores humanos al crear scripts:

- Servicio **logrotate** → Se configura con los siguientes ficheros:

/etc/logrotate.conf
Por defecto, para todos los servicios

```
# rotate log files weekly, monthly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# send errors to root
errors root
# compressed log files
compress
…
```

/etc/logrotate.d/
Sobrescribe logrotate.conf para un servicio concreto

```
/var/log/dpkg.log {
        monthly
        rotate 12
        compress
        notifempty
        create 0664 root
adm
}
```

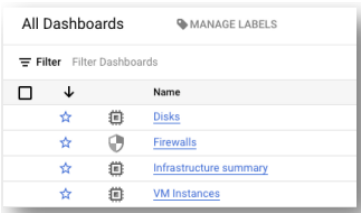
Analizando logs

- Para depuración:
  - Útil para obtener más información cuando algo va mal
  - Activar modo verboso de las aplicaciones
    - P.e. activar flag -d, /etc/init.d/ssh sshd -d
  - Importante: desactivar el modo verboso al volver a producción
- Para monitorización:
  - Problema: abundante información, de la cual mucha puede no ser útil
  - Utilizar herramientas para buscar los mensajes relevantes, p.e.:
    - Swatch: Programa Perl que busca patrones en los logs1
    - LogWatch: Genera resúmenes para su envío por e-mail.
    - Soluciones más complejas, p.e., pila ELK

MONITORIZACIÓN EN GCP

GCP provee un sistema de monitorización para los servicios en uso

- Se accede en la sección “Operaciones”
- Permite obtener métricas en diferentes resoluciones → Minutos, horas, días, … (La mayor resolución es 1 minuto)
- Tiene un coste asociado
- La capa gratuita incluye monitorización básica



Organiza la información en Dashboards de control. Por defecto, incluye varios para los recursos más comunes (Sección Dashboards de “Monitoring”):

Permite crear Dashboard personalizados (con diferentes tipos de gráficos)

Métricas de una VM:

- Por defecto, se obtienen métricas relativas a uso de recursos y red
- Se visualizan en forma de gráfica

Crear alertas que envíen notificaciones cuando sucedan eventos concretos.

- Se gestionan desde la sección “Alertas”
- Para crear alertas personalizadas:
    - Crear una política
    - Elegir la métrica a monitorizar
    - Elegir el umbral de la métrica
    - Elegir el canal de notificación (p.e. e-mail)
    - Revisar y guardar

RENDIMIENTO

Depende del entorno del trabajo, los usuarios van a utilizar una colección de aplicaciones más o menos variada. Es útil caracterizar las máquinas de nuestro entorno para saber cómo van a responder con estas aplicaciones.

Generalmente, una aplicación está limitada por uno de los siguientes:

- Cómputo
    - Operaciones para realizar en la CPU > operaciones con la memoria
    - Ejemplos de aplicaciones:
        - Renderizado de gráficos/video, simulaciones químicas, …
- Memoria
    - Datos para transferir > capacidad de procesado de la CPU
    - Ejemplos de aplicaciones:
        - Análisis de datos, simulaciones de dinámica de fluidos, …

BENCHMARKS

Aplicaciones cuyo objetivo es comprobar el rendimiento de un factor concreto (CPU, memoria, discos, red, librerías, BBDDs, …). Generalmente ejecutan una operación (o pocas) de manera repetida y miden el tiempo necesario. En ocasiones el objetivo es validar que un software es estable.

Características que debe un benchmark debe cumplir:

- Relevante y representativo
- Repetible
- Escalable

SPEC

Standard Performance Evaluation Corporation (SPEC) es un consorcio americano dedicado a desarrollar benchmarks

- SPEC CPU → Software comercial (1.210 US$), última versión: 2017 (anterior 2006). Diferentes versiones: speed, integer, floating point, …
- SPEC Cloud → Software comercial (2.420 US$), última versión: 2018 (anterior 2016)
- Otros: SPEC ACCEL, SPEC MPI,…

FIRESTARTER

Benchmark open-source de stress CPU. Crea y ejecuta diferentes patrones de carga en intervalos configurables por el usuario.

Implementación específica para diferentes arquitecturas:

- Intel: Sandy Bridge, Broadwell, Skylake, Knights Landing, …
- AMD: Zen, Zen+

Algunos resultados para comparar (se utiliza el tipo de instrucción más apropiado para cada chip):

Modelo de CPU	Arquitectura	Lanzamiento	Capacidad	GFLOP/s
Intel Xeon E5-2620	Sandy Bridge	2012	6 cores @ 2.0 GHz	33.9
Intel Xeon E5-2695 v4	Broadwell	2016	18 cores @ 2.1 GHz	397.5
Intel Xeon Gold 6148	Skylake	2017	20 cores @ 2.4 GHz	1000.9
Intel Xeon Silver 4216	Cascade Lake	2019	16 cores @ 2.1 GHz	398.3

STREAM

Benchmark de evaluación de memoria muy popular. Realiza diferentes operaciones con 2 vectores:

- Copy a(i) = b(i)
- Scale a(i) = s*b(i)
- Add a(i) = b(i)+c(i)
- Triad a(i) = b(i)+s*c(i)

El tamaño de los vectores lo define el usuario.

STREAM mide el tiempo en realizar estas operaciones. Es importante compilar y configurar STREAM correctamente para obtener resultados fiables. Algunos resultados para comparar:

Modelo de CPU	Memoria	GB/s	GFLOP/s
Intel Xeon E5-2620	32 GB DDR3 @ 1333 MHz	23.61	33.9
Intel Xeon E5-2695 v4	128 GB DDR4 @ 2400 MHz	48.10	397.5
Intel Xeon Gold 6148	384 GB DDR4 @ 2666 MHz	74.90	1000.9

Para determinar si nuestra aplicación está limitada por cómputo o memoria

Utilizar:

- Herramientas de profiling
- Comerciales: Intel Vtune, ARM MAP, Cray PAT
- Libres: Linux perf, Linux Trace Toolkit, Valgrind, gprof

Modelos de rendimiento

- Descripción matemática que representa una interacción hardware-software
- Generalmente son específicos a una máquina o aplicación
- Requieren de un trabajo de configuración
- A veces, utilizar benchmarks para obtener información del hardware