

# Monitorización

## Administración de Sistemas

Unai Lopez Novoa  
unai.lopez@ehu.eus



Universidad  
del País Vasco

Euskal Herriko  
Unibertsitatea

# Contenido

1. Introducción
2. Gestión de recursos
3. Registros del sistema (logs)
4. Monitorización en GCP
5. Rendimiento



# Introducción

- Para asegurar el buen funcionamiento de una aplicación, hay 4 elementos a observar:

Latencias

Tráfico

Errores

Saturación



# Introducción

- Latencia:
  - El tiempo necesario para completar una petición.
  - Importante distinguir entre peticiones satisfactorias y fallidas.
- Tráfico:
  - Métrica de la demanda de un sistema.
  - Suelen ser específicas de un sistema.
  - Ejemplos:
    - Peticiones HTTP / segundo.
    - Transacciones / segundo.



# Introducción

- 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.



# Introducción

- Cómo observar cada elemento:

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
- *Estos 3 están descritos en el Tema 1, Diapositiva 42*



# Gestión de procesos

- Prioridades de los procesos
  - El planificador del SSOO asigna intervalos de tiempo a los procesos según su prioridad.
  - Esto se controla según 2 valores:
    - Prioridad (**PR**): puede tomar valores en el rango -100 a 39.
    - Valor "nice" (**NI**): puede tomar valores en el rango -20 a 19.
    - *Columnas PR y NI en el comando Top.*
    - *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 SSOO).





# Gestión de procesos

- 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.
  - Los procesos normales ocupan el rango 0-39 de prioridades.
  - Por defecto, el valor "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.



# Gestión de procesos

- Prioridades de los procesos normales
  - Comando **nice**
    - Lanza un comando con un valor Nice concreto
    - Sintaxis: `nice -n valor comando`
      - Valor es relativo (define cuánto 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 Nice.
      - Y cada cambio que haga es irreversible
    - Sintaxis: `renice -n valor -p PID [-g grupo]`
      - Valor es absoluto
    - Ejemplo: `renice -n 15 -p 7552`



# Gestión de procesos

- Prioridades de los procesos en tiempo real
  - Se calcula: **PR** = - 1 - *prioridad\_tiempo\_real*.
  - El valor *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 <prioridad_tiempo_real> <programa>`
  - Ejemplo: `chrt --rr 20 ./miPrograma`
    - Lanzaría ./miPrograma con PR=-21



# Gestión de procesos

- Prioridades de los procesos: comando **ps**

- El comando **ps** puede mostrar datos en diferentes formatos
  - Cada formato puede mostrar una misma prioridad con diferentes valores.

- Formato BSD:

- Comando: **ps al**

```
F UID  PID  PPID  PRI  NI  VSZ  RSS  WCHAN  STAT  TTY  TIME  COMMAND
...
0 1000 2033 1104 21   1  7208 2708  - SN+.  pts/0  0:00  ping www.ehu.eus
```

- Formato Unix:

- Comando: **ps -u unai -o pid,user,pri,nice,args**

```
PID  USER  PRI  NI  COMMAND
...
2033  unai  18   1  ping www.ehu.eus
```

Se muestra un valor de prioridad (PRI) diferente para un mismo proceso (PID=2033).



# Gestión de procesos

- Comando **kill**

- Envía señales a procesos (no sólo matarlos)
- Sintaxis: `kill <opciones> PID`
  - Opciones: `-l`      Mostrar las señales disponibles  
                 `-señal` Mandar una señal al proceso
  - Hay 3 formas de indicar una señal:
    - Con su número    `-19`
    - Con el prefijo SIG `-SIGSTOP`
    - Sin el prefijo SIG `-STOP`
  - Señales:        `-STOP` Parar el proceso  
                     `-CONT` Reanudar el proceso (parado con STOP)  
                     `-KILL` Matar el proceso  
                     ...



# Gestión de procesos

- Comando **ulimit**

- Limitar el uso de recursos
- Los límites sirven para la Shell en uso
- Sintaxis: `ulimit -<opción> [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)

- Ejemplo:

```
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
```



# Gestión de procesos

- 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

- Donde:

- |                  |  |
|------------------|--|
| • usuario/grupo  | Nombre del usuario o grupo (comienza con @)  |
| • tipo-de-límite | soft/hard                                    |
| • ítem           | Puede ser: cpu, nproc, maxlogins, fsize, ... |
| • valor          | Valor para el ítem definido                  |

- Ejemplos:

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

- Mas información: man limits.conf



# Gestión de procesos

- Comando **cpulimit**

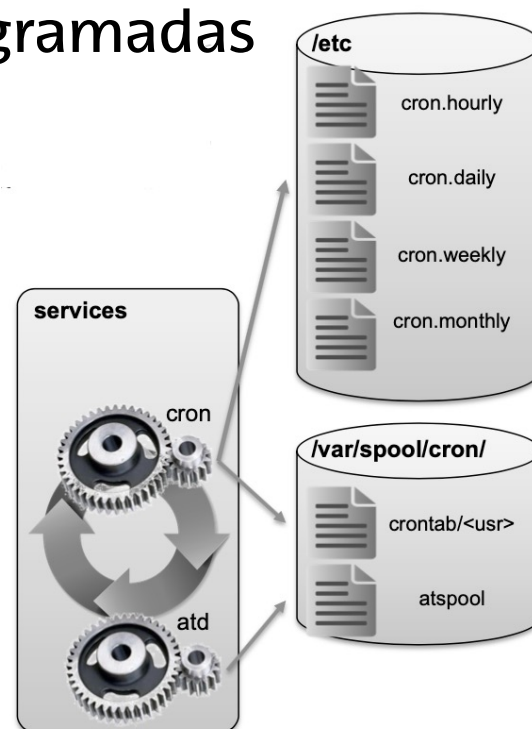
- Permite limitar el % de uso constante de CPU de un proceso
  - ulimit y limits.conf sólo permiten limitar el tiempo total de uso CPU
  - nice y renice permiten reducir la prioridad pero no fijar un umbral
- Está en los repositorios Debian
- Uso: `cpulimit --pid PID --limit <límite>`
  - Donde <límite> es el límite de % CPU máximo que queremos permitir
- Más información:
  - <https://www.tecmint.com/limit-cpu-usage-of-a-process-in-linux-with-cpulimit-tool/>





# Planificación de tareas

- Se pueden programar tareas para que se ejecuten periódicamente (**cron**) o una única vez (**atd**)
- **cron** y **atd** leen periódicamente sus ficheros de configuración para ejecutar tareas programadas
  - Por defecto, cada minuto
- Algunas tareas programables:
  - Rotación de logs
  - Borrar la carpeta /tmp
  - Copias de seguridad
  - Actualizar una BBDD



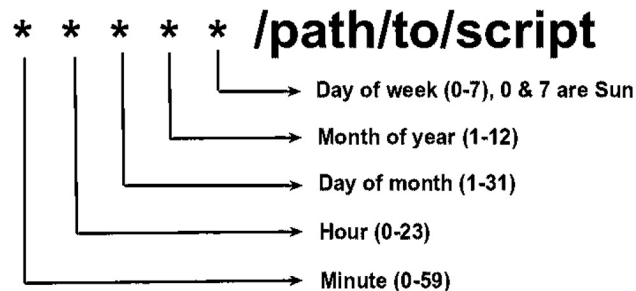
# Planificación de tareas

- Comando **crontab**

- Una línea por tarea programada
- Sintaxis: **crontab** <opciones>
  - Opciones: 

-l	Mostrar las tareas programadas
-e	Editar las tareas programadas
-r	Elimina las tareas programadas

- Cada entrada de cron es una línea sigue la estructura:



# Planificación de tareas

- Comando **crontab**

- Ejemplos:

- `6 17 * * * /scripts/copia.sh` Ejecuta copia.sh los sabados a las 17:00
    - `* * * * * /scripts/miScript.sh` Ejecuta miScript.sh cada minuto
    - `* 5,17 * * * /scripts/api.sh` Ejecuta api.sh a diario a a las 5:00 y 17:00
    - `* */10 * * * /scripts/mon.sh` Ejecuta mon.sh a diario cada 10 minutos

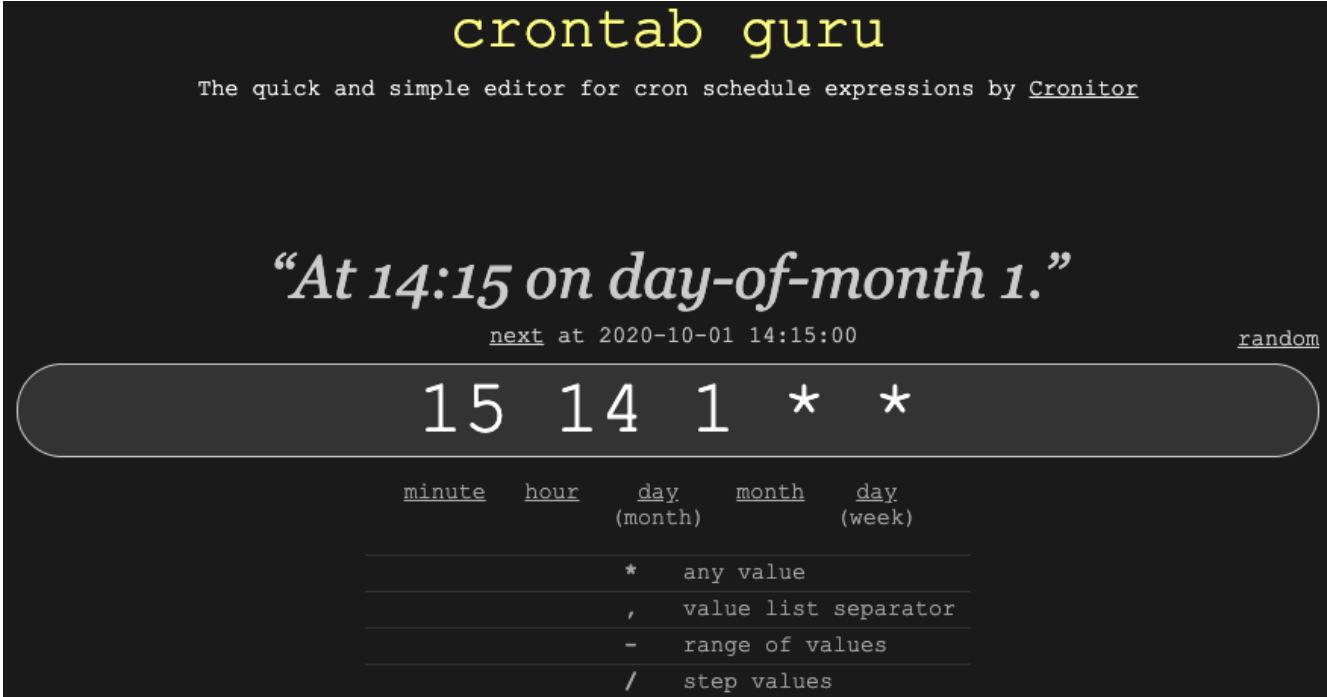
- Más ejemplos en:

- <https://tecadmin.net/crontab-in-linux-with-20-examples-of-cron-schedule/>



# Planificación de tareas

- Comando **crontab**
  - <https://crontab.guru/>
  - Editor online de entradas cron



The screenshot shows the 'crontab guru' website. At the top, it says 'crontab guru' in yellow, followed by 'The quick and simple editor for cron schedule expressions by Cronitor'. Below this, a large quote reads: "At 14:15 on day-of-month 1." Underneath the quote, it says 'next at 2020-10-01 14:15:00' and a 'random' link. A large rounded rectangle contains the cron expression '15 14 1 \* \*'. Below this, a table explains the fields: minute, hour, day (month), month, and day (week). The table lists the symbols used in the cron expression: '\*' for any value, ',' for value list separator, '-' for range of values, and '/' for step values.

<u>minute</u>	<u>hour</u>	<u>day</u> (month)	<u>month</u>	<u>day</u> (week)
		*	any value	
		,	value list separator	
		-	range of values	
		/	step values	



# Planificación de tareas

- **Comando at**

- Controla las tareas a ejecutar por atd
- Para programar una tarea
  - Desde Shell: `at HORA` (donde HORA es una hora en formato HH:MM)
  - Se abre el Shell de at, escribir el/los comando(s) deseado(s):
    - P.e. `ls /home/unai -l`
  - Cerrar la Shell de at (Ctrl + D)
  - La salida estándar (stdout) se envía por mail usando sendmail
    - Conviene revisar `/var/spool/mail/<usuario>`
- Otras opciones:
  - Desde Shell: `at -l` Listado de tareas pendientes
  - Desde Shell: `at -d <ID>` Eliminar tarea (obtener ID con -l)



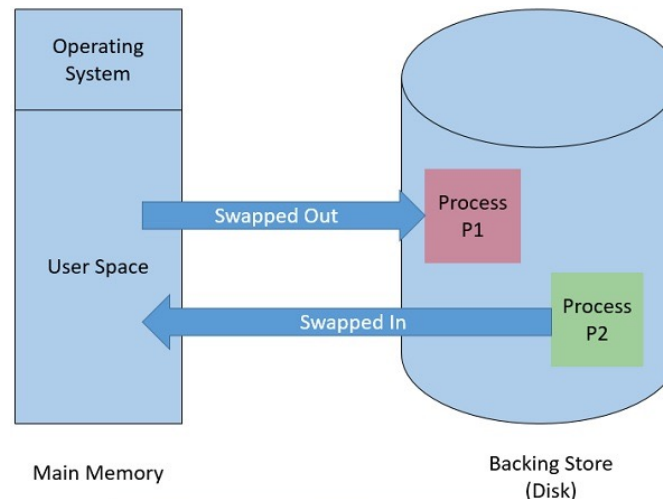
# Ejercicio 1

- Instalar el paquete stress-ng
  - Suite de programas para evaluar el rendimiento
- Ejecutar stress-ng durante 2 minutos con 2 hilos CPU. Mientras está en ejecución, cambiar la prioridad de uno de sus procesos a la mínima posible.
  - ¿Qué sucede?
  - Probar a cambiar la prioridad del mismo a la máxima posible.
- Ejecutar stress-ng durante 3 minutos con 1 hilo CPU. Mientras está en ejecución, limitar su uso de CPU al 50%.
  - Verificar con **top**.



# Gestión de memoria

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



# Gestión de memoria

- La mayoría de sistemas operativos modernos utilizan memoria virtual.
  - Un uso excesivo de Swap puede degradar el rendimiento.
  - Valores referencia de latencias en una jerarquía de memoria<sup>1</sup>:

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





# Gestión de memoria

- Monitorizar la memoria

- Comando **top**

- Utilizar Shift+m 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 (-h para Human readable)

- 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 sin listar todo su contenido



# Gestión de discos y ficheros

- Monitorización

- 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

- 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

NetHogs version 0.8.6-3

PID	USER	PROGRAM	DEV	SENT	RECEIVED
2561	unai	wget	ens4	55.561	84852.891 KB/sec
2312	unai	sshd: unai@pts/2	ens4	1.038	0.296 KB/sec
460	root	/usr/bin/google_osco..	ens4	0.011	0.011 KB/sec
?	root	unknown TCP		0.000	0.000 KB/sec
TOTAL				56.609	84853.198 KB/sec



# Gestión de red

- Monitorización

- 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 -nr <ficheroCaptura>`
    - Se puede añadir el parámetro `-ttt` para incluir la diferencia de tiempo entre cada paquete.



# Gestión de red

- 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>`



# Ejercicio 2

- *Realizar este ejercicio en parejas, seréis A y B*
- A inicia una captura de tráfico con tcpdump en su MV.
  - La captura se debe guardar en un fichero
- B hace ping 10 veces a la máquina virtual de A.
- A para la captura de tráfico.
- Buscad paquetes relacionados con ping en el fichero de captura.
  - ¿Cuántos paquetes encontráis?



# Logs

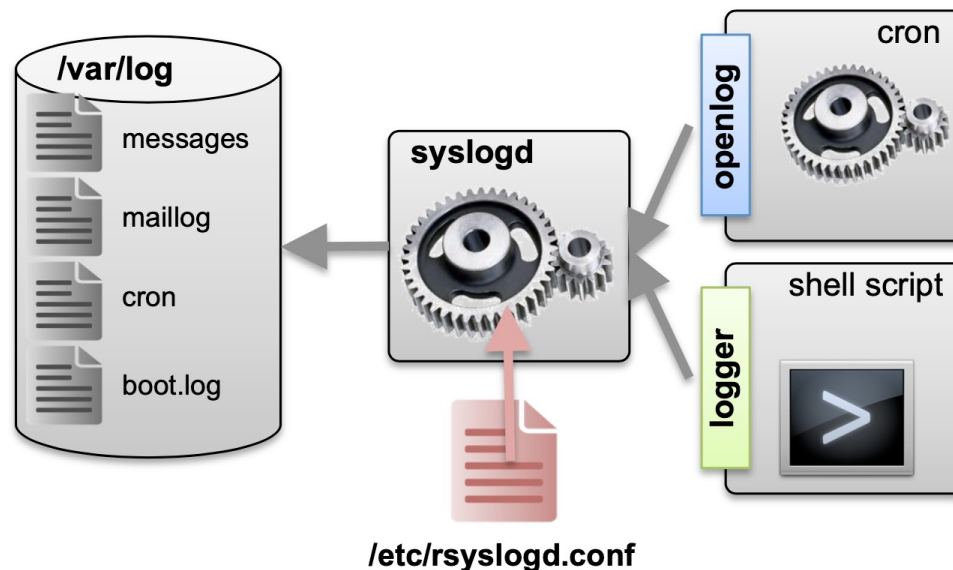
- El kernel de Linux, los servicios y las aplicaciones generan eventos constantemente:
  - Información sobre su estado
  - Información sobre fallos/anomalías
  - Errores de arranque
  - Acceso a información (seguridad)
- Una gestión adecuada de esta información es esencial para descubrir y solucionar problemas
- Todos estos eventos suelen estar gestionados por un único servicio
  - En Unix/Linux es syslog





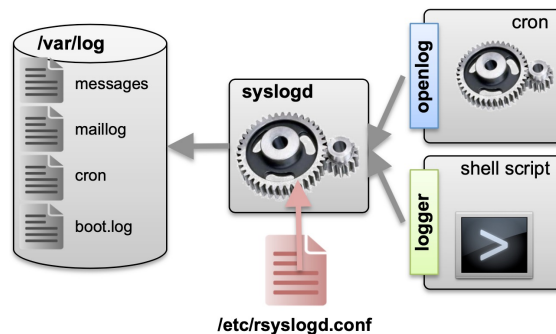
# Syslog

- Es el recolector de eventos empleado por el kernel, servicios y aplicaciones
- Flexible, seguro y fácil de usar
- Está compuesto por los siguientes elementos:



# Syslog

- Partes de syslog:
  - **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.
    - Ver siguiente transparencia.



# Syslog

- **rsyslogd.conf**

- 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
  - P.e.: 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.



# Syslog

- rsyslogd.conf
  - Ejemplo:

```
# 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
...
```

- En Ubuntu, por defecto es: /etc/rsyslog.d/50-default.conf



# Syslog

- 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

- Los logs son una herramienta fundamental para el control y reparación del sistema.

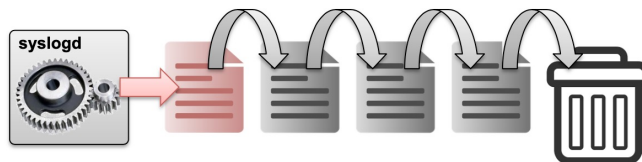
*Pero...*

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



# Gestión de logs

- 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.
  - Ejemplo:

```
#!/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
```

# Gestión de logs

- Rotación de logs
  - 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
}
```





# Gestión de logs

- Analizando logs
  - Para depuración:
    - Útil para obtener más información cuando algo va mal
    - Activar modo verboso de las aplicación
      - 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 logs<sup>1</sup>
      - LogWatch: Genera resúmenes para su envío por e-mail.
      - Soluciones más complejas, p.e., pila ELK.



<sup>1</sup>: Swatch: The Simple Log Watcher: <https://www.linuxtoday.com/blog/swatch-the-simple-log-watcher.html>

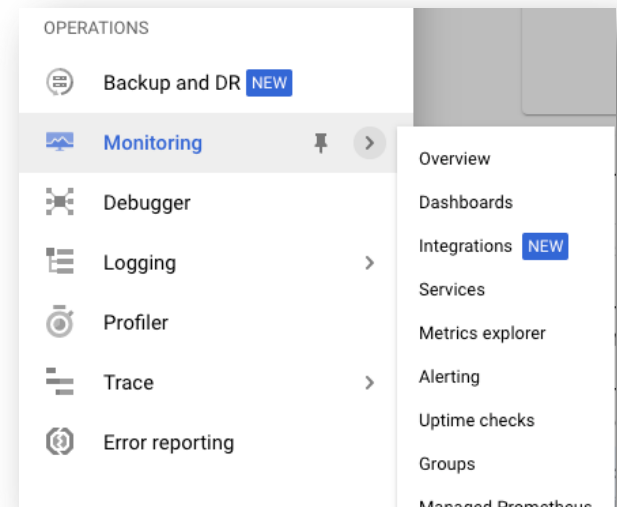
# Ejercicio 3

- Enviar un mensaje al log del sistema desde el terminal y comprobar que se ha añadido correctamente.
- Añadir una regla a syslog para que los mensajes de usuario de tipo "debug" se escriban en `/var/log/user_debug.log`
- Enviar un mensaje de tipo debug y comprobar que se escribe en un `/var/log/user_debug.log`
- Devolver syslog a su situación anterior
  - Eliminar la regla recién creada



# Monitorización en GCP

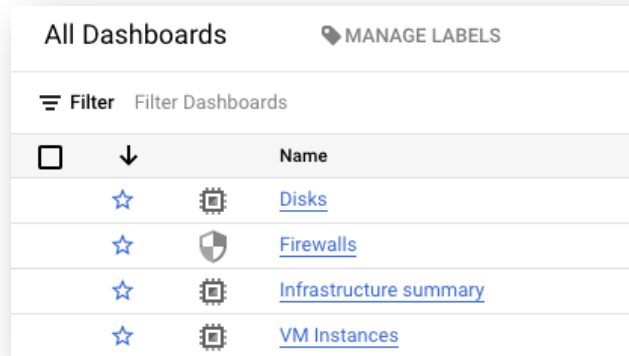
- GCP provee un sistema de monitorización para los servicios en uso
  - Se accede en la sección “Operaciones”
- Permite obtener métricas en diferentes resoluciones
  - Minutos, horas, días, ...
  - La mayor resolución es 1 minuto.
- Tiene un coste asociado
  - La capa gratuita incluye monitorización básica<sup>1</sup>



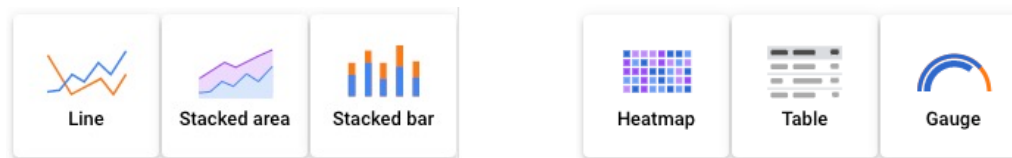
<sup>1</sup>Precios de la suite de operaciones en GCP: <https://cloud.google.com/stackdriver/pricing?hl=es>

# Monitorización en GCP

- Organiza la información en Dashboards de control
  - Por defecto, incluye varios para los recursos más comunes
    - Sección Dashboards de "Monitoring":

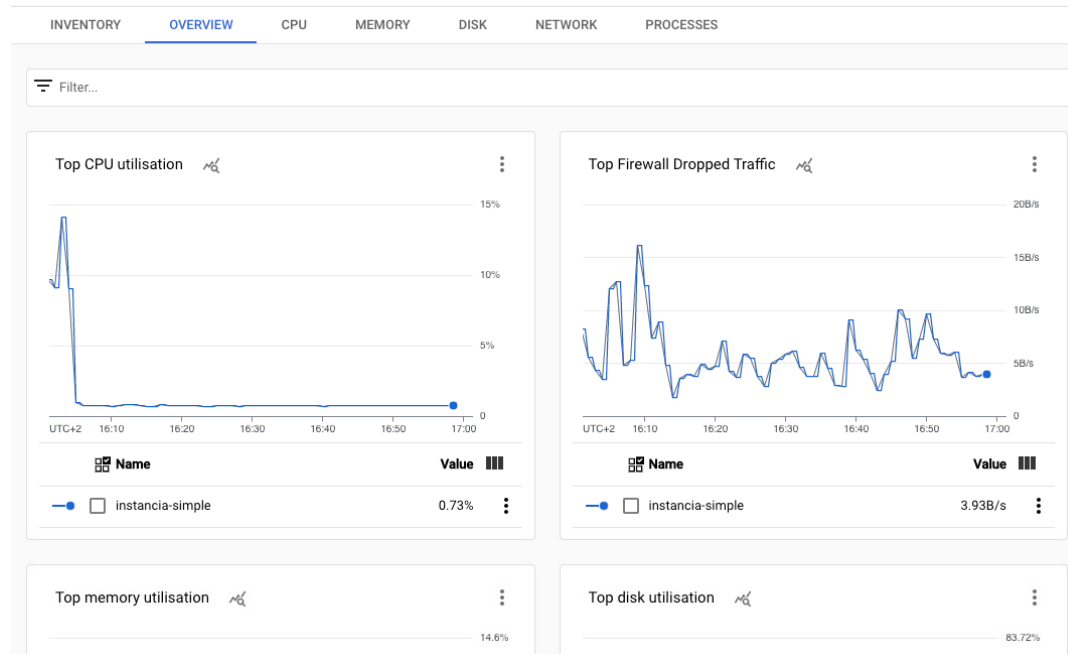


- Permite crear Dashboard personalizados
  - Con diferentes tipos de gráficos:



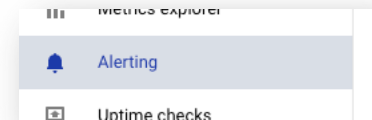
# Monitorización en GCP

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



# Monitorización en GCP

- Se pueden crear alertas que envíen notificaciones cuando sucedan eventos concretos.
  - P.e. cuando una métrica supere un umbral, al detectar un fallo, etc.
  - 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



# Monitorización en GCP

- Monitorizar el estado de Google Cloud
  - Portal de estado: <https://status.cloud.google.com/>
    - Nivel general:

Products	Americas (regions)	Europe (regions)	Asia Pacific (regions)	Middle East (regions)	Multi-regions	Global
Google Cloud Tasks	✓	✓	✓			
Google Compute Engine	✓	✓	✓	✓		
Google Distributed Cloud Edge	✓	✓	✓			
Google Kubernetes Engine	i	i	i	i		
Healthcare and Life Sciences	✓	✓	✓		✓	

- Nivel de región:

Products	europa-west1 Belgium	europa-west2 London	europa-west3 Frankfurt	europa-west4 Netherlands	europa-west6 Zurich	europa-west8 Milan	europa-west9 Paris	europa-west10 Berlin	europa-west12 Turin
Tasks	✓	✓	✓		✓				
Google Compute Engine	✓	✓	✓	✓	✓	✓	✓	✓	✓
Google Distributed	✓	✓	✓	✓	✓	✓			



# Monitorización en GCP

- Monitorizar el estado de Google Cloud
  - Portal de resumen de incidencias recientes por servicio:  
<https://status.cloud.google.com/summary>
  - Incidencias de Google Compute Engine:
    - A fecha 25 de septiembre de 2023.

Google Compute Engine		
SUMMARY	DATE	DURATION
Google Compute Engine is experiencing elevated error rate with diskTypes.aggregatedList API method	16 Sep 2023	✓ 1 hour, 9 minutes
This issue is believed to be affecting a very small number of projects and our Engineering Team is working on it...	13 Sep 2023	✓ 50 minutes
Multiple services for Google Cloud Platform are impacted in us-central1-a	12 Sep 2023	✓ 3 hours, 46 minutes
Google Compute Engine is experiencing intermittent errors in multiple regions	4 Sep 2023	✓ 1 hour, 29 minutes





# Ejercicio 4

- Crear un Dashboard de GCP para monitorizar los siguientes elementos de vuestra MV:
  - Uso (“utilization”) de CPU.
  - N° de bytes recibidos.
- En la MV, descargar la ISO para la última versión de Ubuntu Server.
  - URL: <https://releases.ubuntu.com/22.04.3/ubuntu-22.04.3-live-server-amd64.iso>
- Verificar el incremento de Bytes recibidos el gráfico.
  - Puede tardar unos segundos en aparecer



# 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.



# Rendimiento

- Generalmente, una aplicación está limitada por uno de los siguientes:
- **Cómputo**
  - Operaciones a realizar en la CPU › operaciones con la memoria
  - Ejemplos de aplicaciones:
    - Renderizado de gráficos/video, simulaciones químicas, ...
- **Memoria**
  - Datos a transferir › capacidad de procesamiento de la CPU
  - Ejemplos de aplicaciones:
    - Análisis de datos, simulaciones de dinámica de fluidos, ...



# Benchmarks

- Son aplicaciones cuyo objetivo es comprobar el rendimiento de un factor concreto, p.e.
  - 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



# Benchmarks

- SPEC
  - Standard Performance Evaluation Corporation (SPEC) es un consorcio americano dedicado a desarrollar benchmarks
    - Web oficial: <https://www.spec.org/benchmarks.html>
  - 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,...



# Benchmarks

- 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+
  - Web oficial: <http://tu-dresden.de/zih/firestarter/>
    - Última versión: 2.0 (Enero 2021)
    - Ejercicio: descargar Firestarter y ejecutar en PC o máquina virtual



# Benchmarks

- Firestarter
  - Algunos resultados para comparar:

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

- Se utiliza el tipo de instrucción más apropiado para cada chip.



# Benchmarks

- 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.
- Web oficial: <https://www.cs.virginia.edu/stream/>





# Benchmarks

- STREAM
  - 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

- Resultados de la prueba Triad con tamaño de array  $2^{25}$ .
- Compilado con GCC v7.3 y flags `-O3` y `-march`



# Rendimiento

- ¿Cómo 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



# Bibliografía

- Pablo Abad Fidalgo, José Ángel Herrero Velasco. "Advanced Linux System Administration", OCW UNICAN, 2018:
  - Topic 8: Resource management
  - Topic 9: Logging
  - Publicado bajo licencia Creative Commons BY-NC-SA 4.0
  - <https://ocw.unican.es/course/view.php?id=38>
- Dan Sullivan. "Google Cloud Associate Cloud Engineer: Get Certified", Udemy, 2022.
  - <https://www.udemy.com/course/google-certified-associate-cloud-engineer-2019-prep-course/>
- Consultados en julio 2020 y septiembre 2023

