

MONITORIZACIÓN DE SISTEMAS Y PROGRAMAS

Usuarios, administradores y diseñadores

*¿Cómo medir el
rendimiento de un
sistema informático?*

MONITORIZACIÓN

La monitorización o monitoreo es una técnica para supervisar, analizar y evaluar el comportamiento y el rendimiento de los sistemas informáticos **que están en funcionamiento.**

- El entorno escogido es el sistema operativo Linux (basado en sistemas Unix), ya que la cantidad de herramientas de monitorización en este sistema es relativamente elevada y su uso está muy extendido entre los administradores de sistemas.



INTRODUCCIÓN

Técnicas de medida:
detección de eventos y muestreo



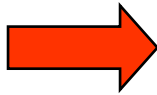
MEDIDA Y MONITOR

La monitorización de sistemas informáticos hace referencia a todo lo relativo a la extracción de información que permita **conocer qué está sucediendo** en ellos.

Hay que decidir qué **datos** hace falta **recoger**, saber dónde se encuentran, y por último, es necesario acceder a ellos **de manera que perturben lo mínimo el funcionamiento del sistema**, para grabarlos, con vistas a su posterior análisis.

LA MEDIDA

Carga



¿Qué está
ocurriendo?



Problema de la medida

- ¿Qué información?
- ¿Dónde está esa información?
- ¿Cómo se puede extraer y dónde grabarla?

El instrumento de medida puede perturbar el funcionamiento del sistema

TÉCNICAS DE MEDIDA

Las técnicas de medida en los sistemas informáticos se basan en **monitorear cambios de estado** en los mismos.

Cada cambio de estado que es **relevante** para la medición de una determinada medida es un **evento**.

Existen cuatro técnicas para monitorear sistemas: **conducidas por eventos**, rastreo (*tracing*), **muestreo** (*sampling*) y medición indirecta.

TÉCNICAS DE MEDIDA

Las técnicas **conducidas por eventos** guardan información suficiente para derivar una medida **solamente cuando ocurren eventos** de interés.

Por lo tanto, el sistema a medir tiene que estar instrumentado (monitoreado) para recoger la información relevante de cada evento.

- Por ejemplo, contar el número de accesos a un disco durante la ejecución de un programa exigiría de un contador en la interrupción de E/S del sistema operativo y volcar ese valor numérico al final de la ejecución del programa.

TÉCNICAS DE MEDIDA

Las técnicas de **rastreo** son similares a las conducidas por eventos, sin **embargo además de los contadores de los eventos de interés se grabarían otras informaciones del estado** del sistema cuando ocurren los eventos para construir una medida derivada.

- Siguiendo el ejemplo anterior, además de contar los accesos a disco, se podría almacenar los nombres/tipos de los ficheros que se acceden.

TÉCNICAS DE MEDIDA

Las técnicas de **muestreo** guardan información relevante del estado del sistema informático a **intervalos regulares**.

Por tanto, el muestreo se convierte en una técnica **estadística** del comportamiento y se pueden perder eventos infrecuentes en el mismo.

- Es lógico que el muestreo se utilice en eventos de muy alta frecuencia donde las cantidades absolutas de su conteo no son importantes sino su significancia estadística.

TÉCNICAS DE MEDIDA

Las técnicas de **medición indirecta** se usan en casos de la medida de interés **no se puede realmente medir observando los eventos**.

- 'Permiten derivar medidas de rendimiento que se pueden calcular a partir de otras medibles, por ejemplo se puede calcular la utilización de un servidor sabiendo el tiempo que se ha utilizado sobre el tiempo total medido.

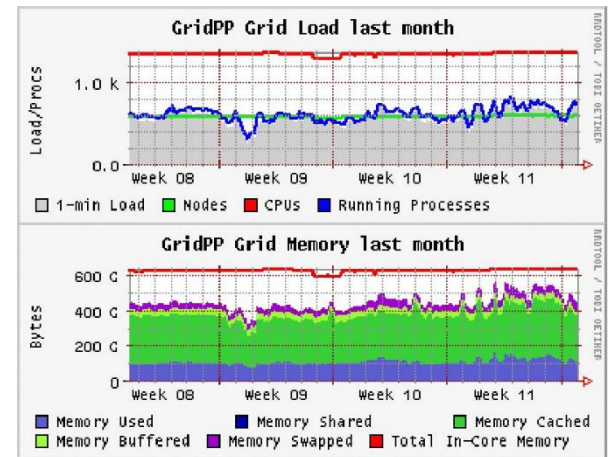
TÉCNICAS DE MEDIDA PRINCIPALES

¿Cómo se toman medidas del sistema?

- Cada vez que ocurre un evento
- Cada periodo fijo de tiempo

Medida por muestreo

Medida por detección de eventos



La mayoría de las herramientas que monitorizan la actividad de los sistemas informáticos emplean una mezcla de las dos: se utilizan contadores de sucesos que se muestren de forma periódica.

DETECCIÓN DE EVENTOS

Estado del sistema

- Contenido de todas las memorias

Evento

- Provoca un cambio del estado

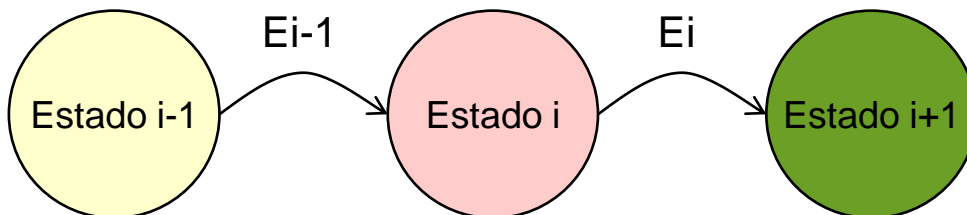
Volumen de información recogida

- Depende de la frecuencia de los eventos

Ejemplos de eventos:

- Inicio/fin de la ejecución de un programa
- Activación de las señales RD* y WR* de memoria
- Acierto/fallo en memoria cache
- Atención a un dispositivo periférico
- Abrir/cerrar un fichero
- ...

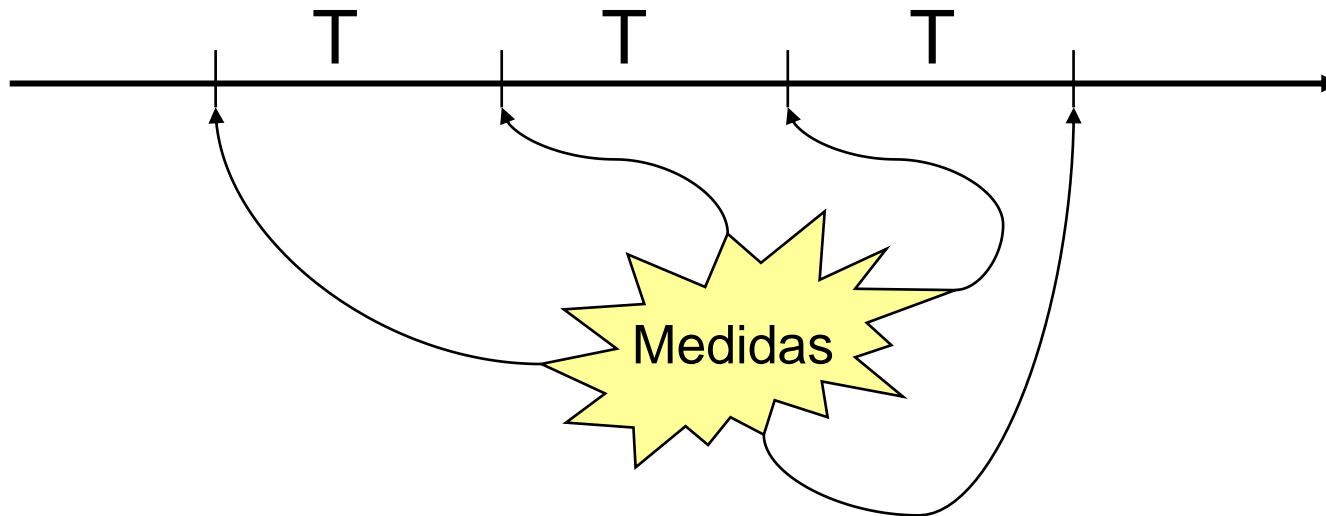
Una gran parte de los eventos (no todos) pueden ser detectados por software



MUESTREO

Observación a intervalos regulares o aleatorios


- Análisis estadístico de datos más fácil
- Volumen de información recogida y precisión: dependen de T





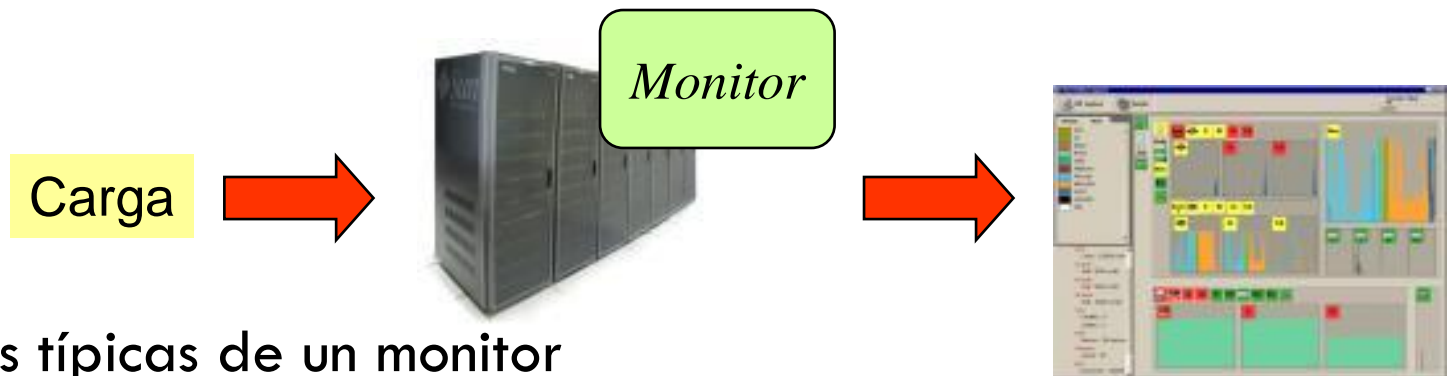
MONITORES DE ACTIVIDAD

Monitores software, hardware e híbridos



CONCEPTO DE MONITOR

Herramienta diseñada para observar la actividad de un sistema informático mientras es utilizado por los usuarios



Acciones típicas de un monitor

- Observar el comportamiento
- Recoger datos estadísticos
- Analizar estos datos
- Mostrar los resultados

UTILIDAD DE LOS MONITORES

Administrador

- Conocer la utilización de los recursos (detección de cuellos de botella)
- Ajustar los parámetros del sistema (sintonización)

Analista

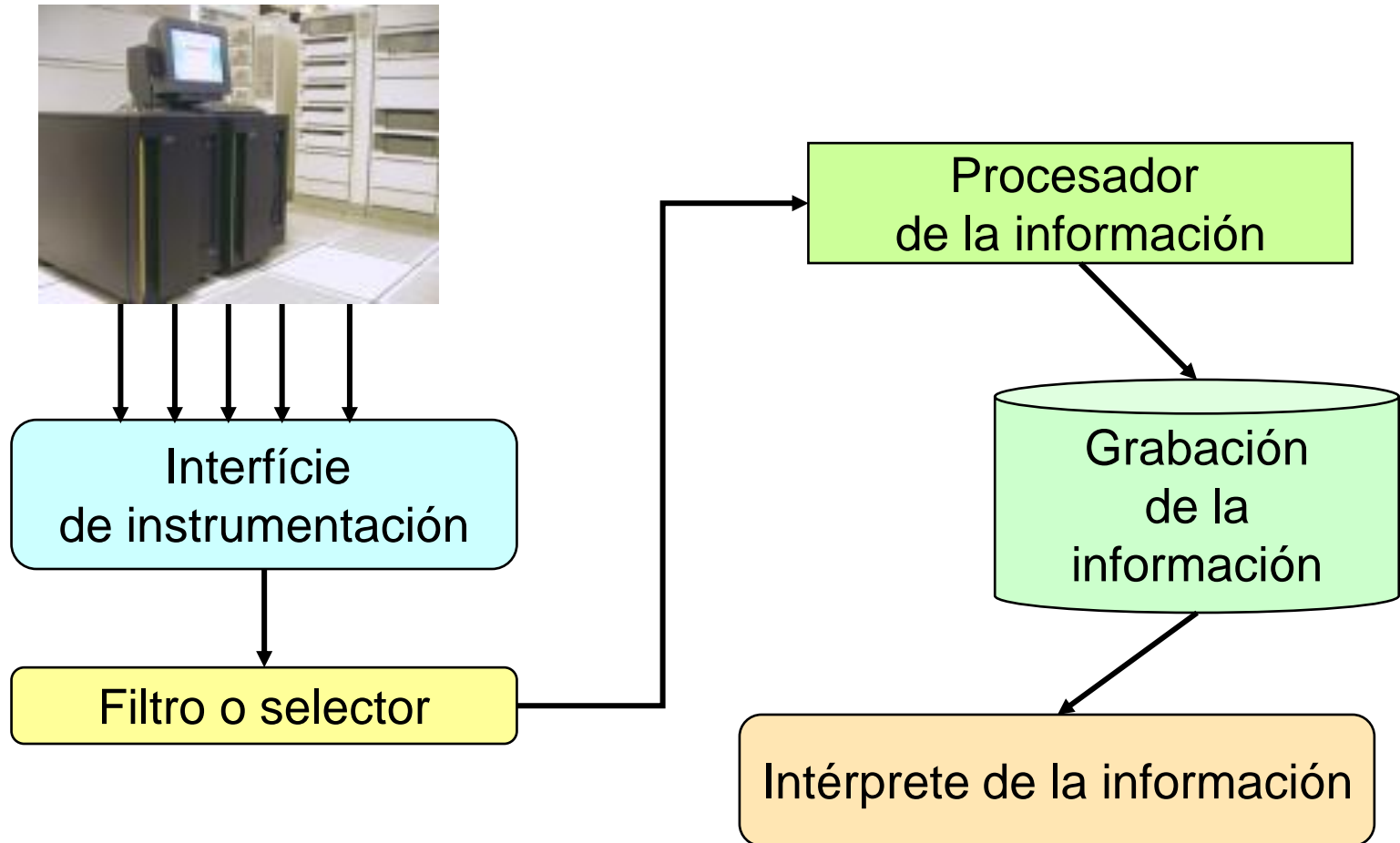
- Parametrizar la carga real
- Calcular los parámetros de entrada a modelos del sistema (analíticos o simulación)

Sistema

- Adaptarse dinámicamente a la carga



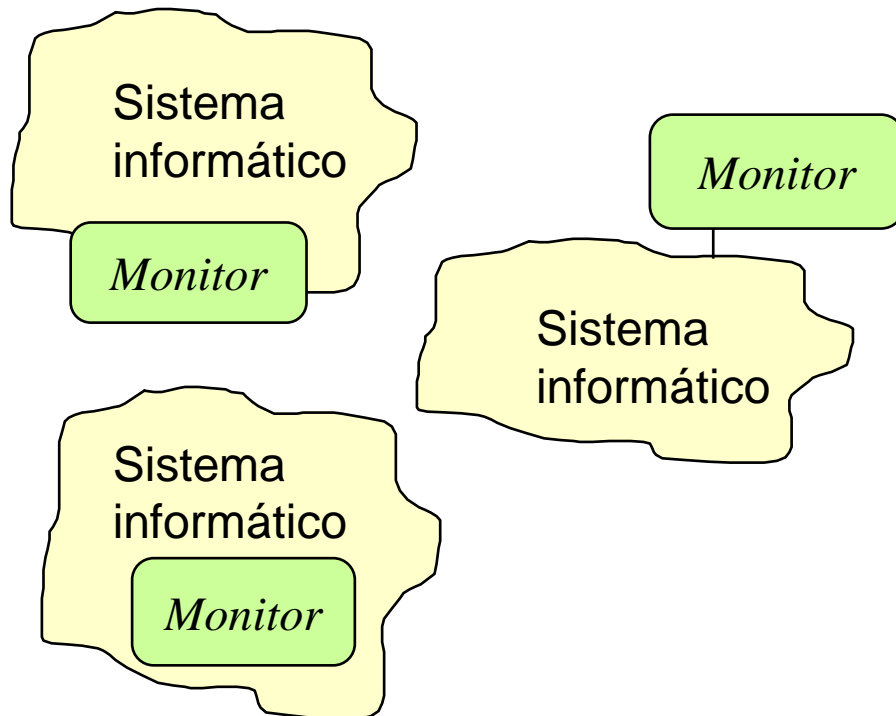
ESQUEMA CONCEPTUAL DE UN MONITOR



ATRIBUTOS DE LOS MONITORES

Interferencia o sobrecarga
(*overhead*)

- Diferentes grados de intrusismo



Precisión

- Calidad de la medida

Resolución

- Frecuencia de medida

Ámbito o dominio de medida

- Qué mide

Anchura

- Bits de información

Capacidad de síntesis de datos

Coste

Facilidad de instalación y uso

IMPLEMENTACIÓN DE LOS MONITORES

Software

- Programas instalados en el sistema

Hardware

- Dispositivos externos al sistema

Híbridos

- Utiliza los dos tipos anteriores

Los más
habituales

Entornos
muy
específicos

MONITORES HARDWARE

Instrumentos independientes (externos) del sistema a monitorizar conectados a este mediante sondas electromagnéticas

Ventajas

- No usan recursos del sistema monitorizado
- Rapidez (circuitos electrónicos)

Inconvenientes

- Los sistemas no facilitan la instalación de sondas
- Personal especializado para su operación
- Hay magnitudes no accesibles por hardware
- Posibles perturbaciones electromagnéticas



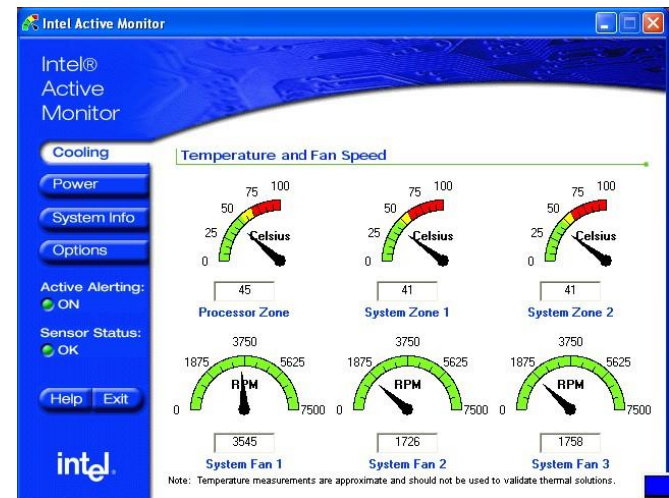
MONITORES SOFTWARE

Son los más usados

Activación → Ejecución de instrucciones → Sobrecarga

Implementación

- Adición de un nuevo programa
- Modificación del software a medir
- Modificación del sistema operativo



MONITORES SOFTWARE

El uso de programas aporta una gran **flexibilidad** en el proceso de monitorización, aunque se paga el precio de un grado de **intrusión** más elevado que en el caso de los monitores hardware.

Dado que la toma de medidas por un monitor software implicará la ejecución de un programa, implicará la **perturbación** del sistema sobre el que se realiza la medida.

El grado de distorsión en la medida suele estimarse mediante una variable denominada **sobrecarga** del monitor (**overhead**).

SOBRECARGA EN UN MONITOR SOFTWARE

La ejecución de las instrucciones del monitor se lleva a cabo en el procesador del sistema monitorizado

$$\text{Sobrecarga} = \frac{\text{Tiempo de ejecución del monitor}}{\text{Intervalo de medida}}$$

Ejemplo de cálculo

- El monitor se activa cada 5 s y cada activación del mismo usa el procesador durante 6 ms

$$\text{Sobrecarga} = \frac{6 \times 10^{-3} \text{ s}}{5 \text{ s}} = 0.0012 = 0.12\%$$

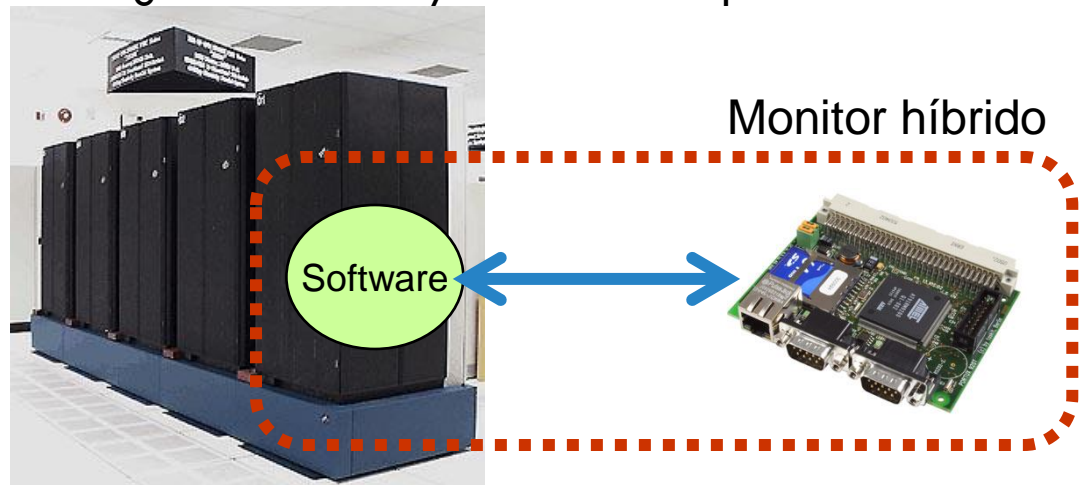
MONITORES HÍBRIDOS

Parte hardware

- Actúa como un dispositivo de I/O que guarda, analiza y procesa la información enviada por la parte software

Parte software

- Código añadido al SO: instrucciones especiales de I/O
- Actúa como una sonda que recoge información y la envía a la parte hardware





MONITORIZAR SERVIDORES

ACTIVIDAD TEMA 2 (OBLIGATORIA)

Supongamos que se activa un monitor de detección de eventos y se detectan los eventos que se producen aleatoriamente $E_t, E_{t+1}, E_{t+2}, \dots, E_{t+n}, \dots$ en los tiempos correspondientes $t, t+1, t+2, \dots, t+n, \dots$

Se quiere sustituir el monitor de detección de eventos por uno de muestreo que hace la misma actividad de monitorización porque el monitor de detección de eventos produce mucha sobrecarga.

El monitor de muestreo se activa cada periodo de tiempo T fijo, pero ajustable.

¿Cómo ajustarías el valor de ese tiempo T de muestreo para que el monitor de muestreo detecte el mismo número de eventos, en media, que el monitor de detección de eventos?

Pista: dibujar ejes de tiempos



MONITORIZAR SERVIDORES

ACTIVIDAD TEMA 2 (VOLUNTARIA)

En la actividad anterior, para determinar una comparativa de sobrecargas entre los monitores de detección de eventos y de muestreo, se ha calculado el incremento entre ambas sobrecargas, As.

Recordemos que el monitor de muestreo se activa cada periodo de tiempo T fijo, pero ajustable.

¿Cómo ajustarías el valor de ese tiempo T de muestreo para asegurar que la sobrecarga del monitor de muestreo sea menor o igual que la del monitor de detección de eventos?

Pista: definición de sobrecarga de monitores



MONITORES SOFTWARE

MONITORIZACIÓN DE SISTEMAS INFORMÁTICOS

"It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suits facts."
- Sir Arthur Conan Doyle

MONITORIZACIÓN EN LINUX


Herramientas de medida:

`time, who, w,
uptime, ps, top,
vmstat, df, du,
hdparm, sar, mpstat,
iostat`

TIME

Mide el tiempo de ejecución de un programa

- Refleja la percepción de las prestaciones del sistema por parte del usuario
 - `real`: tiempo total usado por el sistema (tiempo de respuesta)
 - `user`: tiempo de CPU ejecutando en modo usuario (`user-state CPU time`)
 - `sys`: tiempo de CPU en modo supervisor (`system-state CPU time`) ejecutando código del núcleo



```
%time quicksort
real      6m 23s
user      3m 50s
sys       2m 10s
```

Tiempo de respuesta = `real` = 383 s

Tiempo de CPU = `user+sys` = 360 s (94% del total)

Tiempo de espera = `real-(user+sys)` = 23 s (6% del total)

consumido en espera de I/O o en la ejecución de otros programas

WHO Y W

- who: quién está conectado al sistema (*logged on*)

```
fede      :0      Oct 30 15:07 (console)
xavi      pts/0    Oct 30 17:45
(paraíso.disca.upv.es)
```

- w: quién está conectado al sistema (*logged on*) y qué hace

```
% w
  1:38pm  up  4:27, 18 users,  load average: 0.04, 0.03, 0.04
USER      TTY      FROM          LOGIN@      IDLE        JCPU        PCPU        WHAT
jaume     tty1      kaizen.gap.upv.e  9:17am    2:02m    2:48      0.48s    -sh
fede      tty2                10:28am   51:02     0.14s    0.03s    rlogin ma
xavi      :0        songoku.disca.up  1:20pm     ?         7:32      ?         -
pperez    tty3                10:02am   29:22     0.18s    0.14s    ssh tiberio.
```

CARGA MEDIA DEL SISTEMA UNIX

- Carga media (*system load average*): número medio de procesos en la cola del núcleo
 - No esperan ningún evento externo (p.e. pulsar una tecla)
 - No ha ejecutado ninguna instrucción de espera (p.e. `wait`)
 - No está parado (p.e., no lo han parado con CTRL-Z)
- Fuentes de inexactitud de esta medida
 - Cuenta como ejecutables los que esperan I/O basada en disco, incluyendo discos montados con NFS (*network file system*)
 - No hace distinción según las prioridades de los procesos

UPTIME

Tiempo que lleva el sistema en marcha y la carga media que soporta

```
% uptime
  1:21pm  up  1 day, 4:09, 18 users,  load average: 1.04,
0.30, 0.09
```

Hora actual

Tiempo en marcha

Último minuto

5 últimos minutos

15 últimos minutos

- Estimación de la carga
 - Operación normal: hasta 3
 - Muy alta: entre 4 y 7
 - Excesivamente alta: mayor que 10
- La carga se tolera según la configuración de cada sistema

PS (PROCESS STATUS)

Información sobre el estado de los procesos del sistema

- Es una de las herramientas más importantes empleadas en tareas de monitorización
- Tiene una gran cantidad de parámetros

```
$ ps aur
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME
COMMAND
miguel      29951  55.9   0.1  1448    384 pts/0    R       09:16   0:11
tetris
carlos      29968  50.6   0.1  1448    384 pts/0    R       09:32   0:05
tetris
xavier      30023   0.0   0.5  2464   1492 pts/0    R       09:27   0:00
ps aur
```

INFORMACIÓN APORTADA POR *PS*

- USER
 - Usuario que lanzó el proceso
- %CPU, %MEM
 - Porcentaje de procesador y memoria física usada
- SIZE (○ VSIZE)
 - Memoria (KB) de datos (no código) ocupada por el proceso (*non shared virtual memory*)
- RSS (*resident size*)
 - Memoria (KB) física ocupada por el proceso
- STAT
 - R (*runnable*), T (*stopped*), P (*waiting for page-in*), D (*waiting for disk I/O*), S (*sleeping for less than 20 s*), I (*idle for more than 20 s*), Z (*zombie: terminated but not died*)
 - W (*swapped out*), > (*memory soft limit exceeded*)
 - N (*running niced*), < (*high niced level*)

TOP

- Carga media, procesos, consumo de memoria
- Se actualiza dinámicamente

```
8:48am up 70 days, 21:36, 1 user, load average: 0.28, 0.06, 0.02
47 processes: 44 sleeping, 3 running, 0 zombie, 0 stopped
CPU states: 99.6% user, 0.3% system, 0.0% nice, 0.0% idle
Mem: 256464K av, 234008K used, 22456K free, 0K shrd, 13784K buff
Swap: 136512K av, 4356K used, 132156K free 5240K cached
```

PID	USER	PRI	NI	SIZE	RSS	SHARE	STAT	LC	%CPU	%MEM	TIME	COMMAND
9826	carlos	0	0	388	388	308	R	0	99.6	0.1	0:22	simulador
9831	miguel	19	0	976	976	776	R	0	0.3	0.3	0:00	top
1	root	20	0	76	64	44	S	0	0.0	0.0	0:03	init
2	root	20	0	0	0	0	SW	0	0.0	0.0	0:00	keventd
4	root	20	19	0	0	0	SWN	0	0.0	0.0	0:00	ksoftiq
5	root	20	0	0	0	0	SW	0	0.0	0.0	0:13	kswapd
6	root	2	0	0	0	0	SW	0	0.0	0.0	0:00	bdfush
7	root	20	0	0	0	0	SW	0	0.0	0.0	0:10	kdated
8	root	20	0	0	0	0	SW	0	0.0	0.0	0:01	kinoded
11	root	0	-20	0	0	0	SW<	0	0.0	0.0	0:00	recovered

VMSTAT (VIRTUAL MEMORY STATISTICS)

Paging (paginación), *swapping*, interrupciones, cpu

- La primera línea no sirve para nada

```
% vmstat -n 1 6
```

procs			memory				swap		io		system		cpu		
r	b	w	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id
0	0	0	868	8964	60140	342748	0	0	23	7	222	199	1	4	95
0	0	0	868	8964	60140	342748	0	0	0	14	283	278	0	7	93
0	0	0	868	8964	60140	342748	0	0	0	0	218	212	6	2	93
0	0	0	868	8964	60140	342748	0	0	0	0	175	166	3	3	94
0	0	0	868	8964	60140	342752	0	0	0	2	182	196	0	7	93
0	0	0	868	8968	60140	342748	0	0	0	18	168	175	3	8	89

- Procesos: *r* (*runnable*), *b* (*I/O blocked*), *w* (*swapped out*)
- Bloques por segundo transmitidos: *bi* (*blocks in*), (*blocks out*)
- KB/s entre memoria y disco: *si* (*swapped in*), *so* (*swapped out*)
- *in* (*interrupts por second*), *cs* (*context switches*)

INFORMACIÓN SOBRE LOS DISCOS

- `df` (*filesystem disk space usage*)

```
$ df
Filesystem            1k-blocks      Used Available Use% Mounted
on
/dev/hda2              9606112    3017324    6100816   34% /
/dev/hdb1             12775180    9236405    3140445   75% /home
```

- `du` (*file space usage*)
- `hdparm` (*hard disk parameters*)

```
$ du doc
160      doc/cartas
432      doc
```

```
$ hdparm -g /dev/hda
/dev/hda:
geometry      = 790/255/63, sectors = 12706470, start = 0
```


```
$ hdparm -tT /dev/hda
Timing buffer-cache reads:    128 MB in  1.15 seconds =111.30
MB/sec
Timing buffered disk reads:   64 MB in  6.04 seconds = 10.60
MB/sec
```

EL DIRECTORIO /PROC

- Contiene ficheros con información del sistema
 - Configuración
 - Estadísticas: contadores

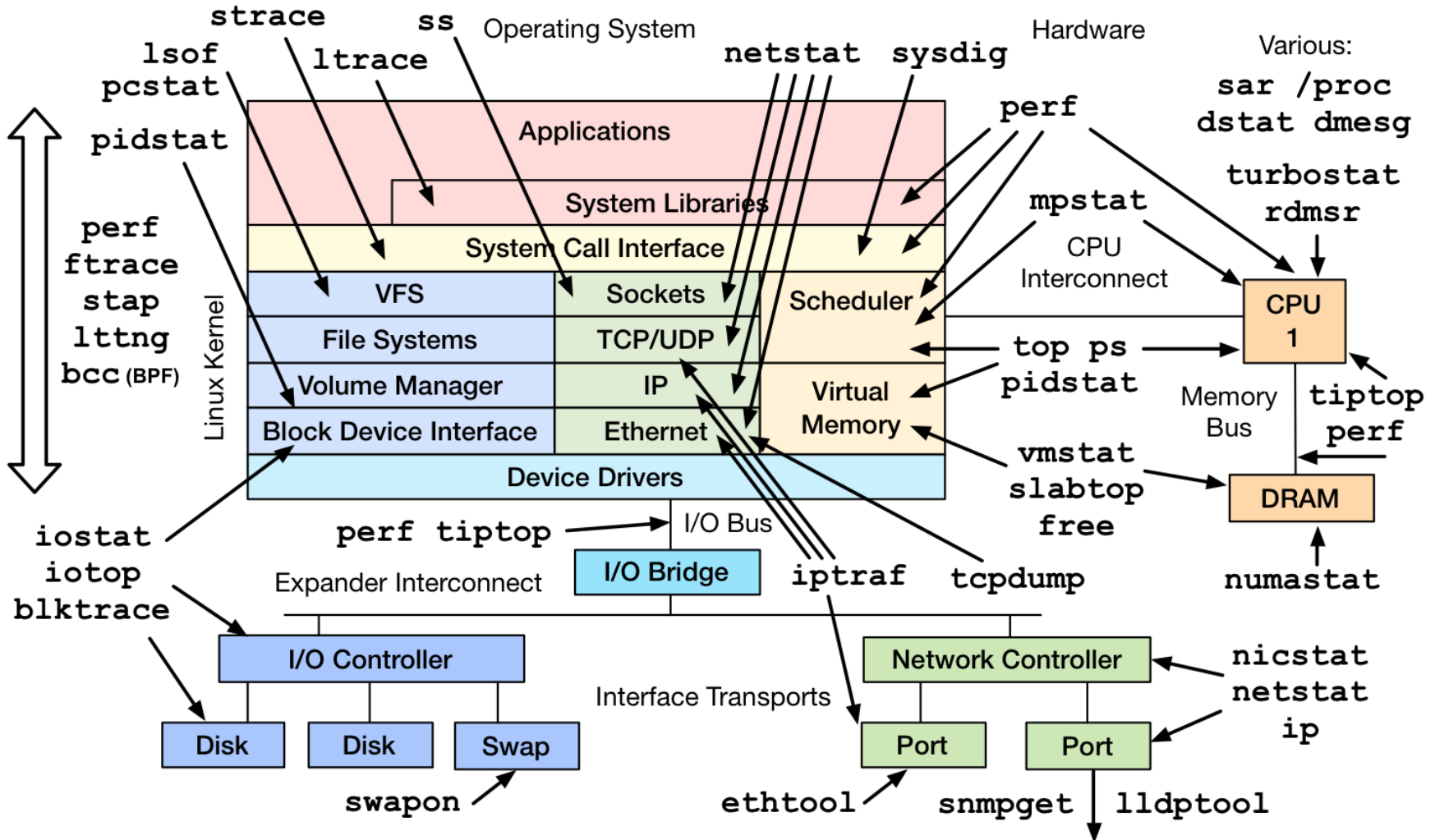
- Ejemplos

- cpuinfo
- meminfo
- interrupts
- devices
- etc.



```
%more /proc/cpuinfo
processor          : 0
vendor_id         : GenuineIntel
cpu family        : 6
modelo            : 5
modelo name       : Pentium II (Deschutes)
stepping          : 2
cpu MHz           : 350.807487
cache size        : 512 KB
fdiv_bug          : no
fpu               : yes
fpu_exception     : yes
cpuid level       : 2
wp               : yes
flags             : fpu vme de pse tsc msr
bogomips          : 349.80
```

Linux Performance Observability Tools



<http://www.brendangregg.com/linuxperf.html> 2017

ERRORES TÍPICOS

1. Falta de propósito en el proceso.
2. No PENSAR qué variables se necesitan para dar solución al problema (1).
3. Medir más variables de las necesarias generando una gran cantidad INÚTIL de datos.
4. Creer que por monitorizar más variables y más tiempo, será mayor el resultado.

MONITOR TOP

Uso de la CPU

```
$top -b -d1 -n1 | grep -i "%Cpu(s)"
```

```
>%Cpu(s): 3,0 usuario, 0,7 sist, 0,9 adecuado, 93,8  
          inact, 1,7 en espera, 0,
```

Uso de la memoria principal

```
$top -b -d1 -n1 | grep -i "KiB Mem"
```

```
>KiB Mem : 7921748 total, 4593252  
free, 1231160 used, 2097336 buff/cache
```

MONITOR TOP

Información sobre un proceso concreto

```
$top -b -n 1 -p <PID> | tail -2
```

```
> PID USUARIO PR NI VIRT RES SHR S %CPU  
%MEM HORA+ ORDEN  
16899 acsic 20 0 44528 3988 3184 S 0,0 0,1 0:02.77  
top
```

MONITOR VMSTAT

```
$ vmstat 1 10
```

```
procs -----memory----- ---swap-- -----io----- -system-- -----cpu-----
r b  swpd  libre búfer caché  si  so  bi  bo  in  cs us sy id wa st
19 0    0 5775832 49840 916784  0  0  61   8  50 216  4  0 95  1  0
16 0    0 5775588 49840 916784  0  0   0   0 4205 1607 100 0 0 0 0
16 0    0 5775588 49840 916784  0  0   0   0 4114 1968 100 0 0 0 0
17 0    0 5775588 49840 916784  0  0   0   0 4166 1428 100 0 0 0 0
16 0    0 5775620 49840 916784  0  0   0   4 4159 1815 100 0 0 0 0
16 0    0 5775588 49840 916784  0  0   0   0 4166 1811 100 0 0 0 0
17 0    0 5775556 49840 916784  0  0   0   0 4196 1563 100 0 0 0 0
16 0    0 5775556 49840 916784  0  0   0   0 4160 1936 100 0 0 0 0
16 0    0 5775588 49840 916784  0  0   0   0 4128 1792 100 0 0 0 0
17 0    0 5775604 49844 916784  0  0   0 224 4177 1447 100 0 0 0 0
```

```
$ vmstat 1 2 | tail -<fila> | awk '{print $<columna>}'
```