sysstat Performance statistics

Kernel counters → sampled by → **sadc** collector → writes → **/var/log/sa/saDD** daily file → read by → **sar -f** report

CONCEPTS

Give every command an interval (seconds) and a count; without them you only get an average since boot.

The first iostat report covers the time since boot; read the later ones, or add -y to drop it.

sar history only exists if the sadc collector runs (cron job or timer from the sysstat package).

High %util means saturation only on one-request-at-a-time disks; SSDs and RAID can stay busy yet keep up.

DISKS (IOSTAT)

iostat -x 2
Extended disk stats every 2 s

iostat -xz 2
Only devices with activity

iostat -c 2
CPU report only

iostat -xy 1 5
Five samples, no since-boot report

DISKS (IOSTAT) (cont.)

iostat -xm 2
Rates in MB per second

iostat -d -h
Human-readable device report

iostat -p *<device>* **2**
Device with its partitions

iostat -t -x 5
Timestamp every report

SYSTEM (SAR LIVE)

sar -u 2 5
CPU utilisation, 5 samples

sar -r 2 5
Memory utilisation

sar -n DEV 2 5
Network throughput per interface

sar -d -p 2 5
Per-disk activity, readable names

sar -q 2 5
Load average and run queue

sar -P ALL 2 5
Each CPU core

sar -S 2 5
Swap usage

sar -n EDEV 2 5
Interface errors and drops

SYSTEM (SAR LIVE) (cont.)

sar -q PSI 2 5
Pressure stall (CPU, I/O, memory)

HISTORY (SAR -F)

sar -u
Today's CPU history

sar -r -f /var/log/sa/sa*DD*
Memory on day DD of the month

sar -u -f /var/log/sa/sa*DD* **-s 09:00 -e 11:00**
CPU in a time window

sar -q -1
Load history from yesterday

PROCESSES (PIDSTAT)

pidstat -u 2 5
CPU per process

pidstat -r 2 5
Memory and page faults per process

pidstat -d 2 5
Disk I/O per process

pidstat -p *<pid>* **1**
Watch one process

pidstat -t -p *<pid>* **1**
Per-thread numbers

PROCESSES (PIDSTAT) (cont.)

pidstat -w 2 5
Context switches per process

pidstat -l 2 5
Full command line

pidstat -C *<name>* **2**
Only commands matching name

READING THE NUMBERS

iostat -x columns

Column	Meaning
`%util`	time with I/O in flight
`await`	avg ms per request, incl. queue
`r_await / w_await`	same, reads / writes
`aqu-sz`	average queue length
`rareq-sz / wareq-sz`	avg request size, KiB
`%iowait`	CPU idle, I/O pending
`%steal`	CPU lost to hypervisor

READING THE NUMBERS (cont.)

pidstat counters

Column	Meaning
`cswch/s`	voluntary switches: blocked, waiting
`nvcswch/s`	involuntary: preempted by others
`majflt/s`	faults that read from disk
`minflt/s`	faults served from memory
`kB_ccwr/s`	writes cancelled before disk

Rising await together with a growing aqu-sz points at a saturated device; high %iowait with an idle CPU says the same.

RECIPES

Is the disk the bottleneck?

```
iostat -xz 2
pidstat -d 2   # who is doing the I/O
```

What happened at 3 am?

```
sar -u -f /var/log/sa/sa<DD> -s 02:30 -e 03:30
sar -q -f /var/log/sa/sa<DD> -s 02:30 -e 03:30
```

HELP

man sar
Manual page with all report fields