Linux System Monitoring

Important Notice : Courseware - Legal

This courseware is both the product of the author and of freely available opensource and/or public domain materials. Wherever external material has been shown, it's source and ownership have been clearly attributed. We acknowledge all copyrights and trademarks of the respective owners.

The contents of the **courseware PDFs are considered proprietary** and thus cannot be copied or reproduced in any form whatsoever without the explicit written consent of the author.

Only the programs - **source code** and binaries (where applicable) - that form part of this courseware, and that are made available to the participant, are released under the terms of the permissive **MIT license**.
Under the terms of the MIT License, you can certainly use the source code provided here; you must just attribute the original source (author of this courseware and/or other copyright/trademark holders).

VERY IMPORTANT :: Before using this source(s) in your project(s), you *MUST* check with your organization's legal staff that it is appropriate to do so.

The courseware PDFs are *not* under the MIT License, they are to be kept confidential, non-distributable without consent, for your private internal use only.

The duration, contents, content matter, programs, etc. contained in this courseware and companion participant VM are subject to change at any point in time without prior notice to individual participants.

Care has been taken in the preparation of this material, but there is no warranty, expressed or implied of any kind, and we can assume no responsibility for any errors or omisions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

2000-2017 Kaiwan N Billimoria
kaiwanTECH, Bangalore, India.

References:

“Linux Performance and Tuning Guidelines”, IBM.

Linux Performance, Brendan Gregg
This page links to various Linux performance material I've created, including the tools maps on the right. These show: Linux observability tools, Linux benchmarking tools, Linux tuning tools, and Linux sar. For more diagrams, see my slide decks below.

Linux Tools for the serious Systems Programmer
Saturday 28 December 2013 05:30 PM
A fairly exhaustive list, with links, of tools (and techniques) one can use for debugging on the Linux OS.

The Different Types of Server Monitoring Software
Here are the various types of server monitoring software and what they specialize in, in regards to different performance metrics.

SO: How to determine CPU and memory consumption from inside a process?

What are we really trying to achieve here?

From Brendan Gregg's blog:

"

Performance Analysis Methodology

A performance analysis methodology is a procedure that you can follow to analyze system or application performance. These generally provide a starting point and then guidance to root cause, or causes. Different methodologies are suited for solving different classes of issues, and you may try more than one before accomplishing your goal.

Analysis without a methodology can become a fishing expedition, where metrics are examined ad hoc, until the issue is found – if it is at all.

Methodologies documented in more detail on this site are:

- The USE Method: for finding resource bottlenecks
- The TSA Method: for analyzing application time
- Off-CPU Analysis: for analyzing any type of thread wait latency
- Active Benchmarking: for accurate and successful benchmarking.

...

USE Method

For every resource, check:

Utilization
Saturation
Errors

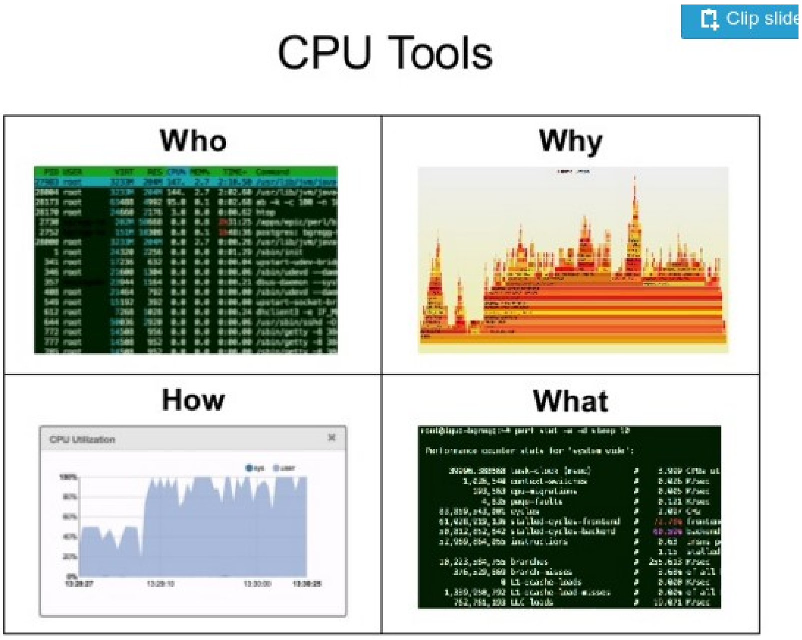
...

"

CPU Workload Characterization

<< Source >>

- Who
- Why
- What
- *How*



CPU Tools

Who

Why

How

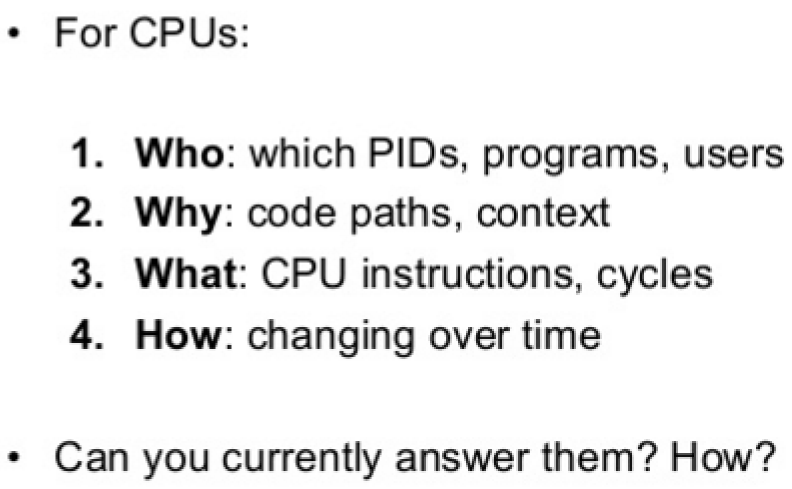
What

The "Who" is usually easy: tools like top, htop, mpstat, perf top, etc will show you.
The "How" is also relatively easy- lots of system monitoring tools available (sar, nagios, cacti, nmon, sysmon, etc).

The problem areas are usually the "Why" and "What"!

- Who
- Why
- What
- How

P.T.O. -->

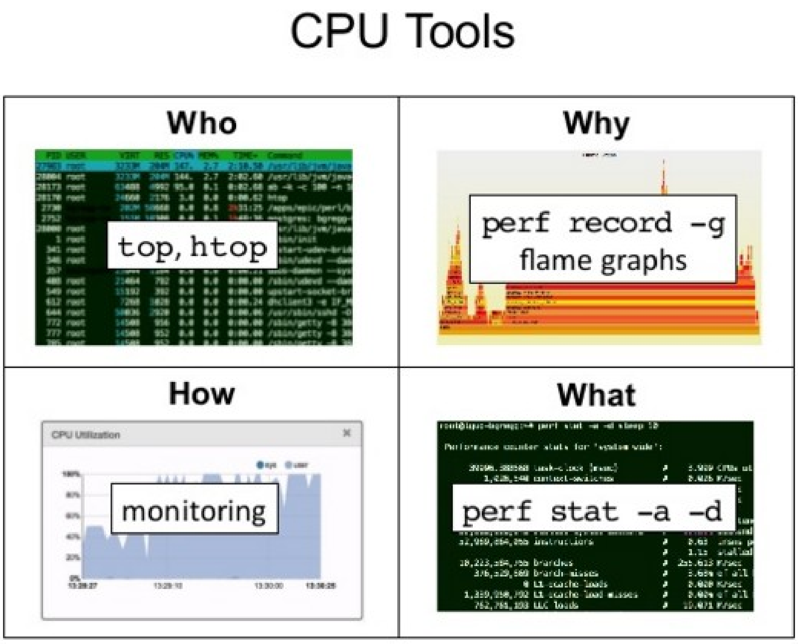


- For CPUs:

  1. **Who**: which PIDs, programs, users
  2. **Why**: code paths, context
  3. **What**: CPU instructions, cycles
  4. **How**: changing over time

- Can you currently answer them? How?

Here's how:



CPU Tools

Who	Why
top, htop	perf record -g flame graphs
How	**What**
monitoring	perf stat -a -d

From Brendan Gregg's blog:

" Broken Linux Performance Tools (SCaLE14x, 2016)

At the Southern California Linux Expo (SCaLE 14x), I gave a talk on Broken Linux Performance Tools. This was a follow-on to my earlier Linux Performance Tools talk originally at SCaLE11x (and more recently at Velocity as a tutorial). This broken tools talk was a tour of common problems with Linux system tools, metrics, statistics, visualizations, measurement overhead, and benchmarks. It also includes advice on how to cope (the green "What You Can Do" slides).
A video of the talk is on youtube and the slides are on slideshare or as a PDF. ..."

Tools for Working with Processes

The participant should try out these utilities, tips, etc on their system.

ps

`man ps` *[ps(1)] for all details of course.*

Tips:

PROCESS FLAGS (usually the 1st field - 'F')

The sum of these values is displayed in the "F" column, which is provided by the flags output specifier:

1 forked but didn't exec
4 used super-user privileges

PROCESS STATE CODES (usually the 2nd field - 'STAT')

Here are the different values that the s, stat and state output specifiers (header "STAT" or "S") will display to describe the state of
a process:

D uninterruptible sleep (usually IO)
R running or runnable (on run queue)
S interruptible sleep (waiting for an event to complete)
T stopped, either by a job control signal or because it is being traced
W paging (not valid since the 2.6.xx kernel)
X dead (should never be seen)
Z defunct ("zombie") process, terminated but not reaped by its parent

For BSD formats and when the stat keyword is used, additional characters may be displayed:

< high-priority (not nice to other users)
N low-priority (nice to other users)
L has pages locked into memory (for real-time and custom IO)
s is a session leader
l is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
+ is in the foreground process group

Also, regarding 'STAT' column:

```
s : is a session-leader
I : multithreaded application (uses CLONE_THREAD)
+ : is in the foreground process group
```

```
ps -l   : see details
ps -L  : see individual threads belonging to a process;
[usermode PID == kernel TGID  (the actual PID for userspace)
 usermode Thread == kernel PID ].

ps fax  : show tree
ps faxj : show tree with more detail

ps -Aefww : show all arguments
```

ps : Customized Output Fields

To see every process with a user-defined format:

```
   ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
   ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
   ps -Ao pid,tt,user,fname,tmout,f,wchan

   ps -eo pid,comm:16,stackp
```

<< comm:16 => 16 chars width; stackp => show stack pointer (see man page for all possible fields that can be shown with ps -eo !) >>

Print only the process IDs of syslogd:

```
   ps -C syslogd -o pid=
```

Print only the name of PID 42:

```
   ps -p 42 -o comm=
```

Also see and try:
pstree
pgrep, pkill
top
lsof : Lists Open Files
pidstat

*See the post "**LINUX Quick Reference Cheat Sheet**" from my blog (covers vmstat, ps, top).*

---

vmstat

VM mode, with repitition:

Repeat interval (sec)

```
# vmstat 5
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0 158496 325136 446320 2282396    0    0     0     7  470 1477  2  1 97  0  0
 0  0 158496 323896 446332 2282788    0    0     3    78  544 1796  4  1 95  1  0
 0  0 158496 324980 446336 2278512    0    0     0    48  523 1758  3  1 96  0  0
 0  0 158492 321732 446336 2278476    6    0    10    23  513 1807  2  1 96  0  0
...
```

System mostly idle..

<<

Procs
r : The number of runnable processes (running or waiting for run time)
b : The number of processes in uninterruptible sleep

Swap
si: Amount of memory swapped in from disk (/s).
so: Amount of memory swapped to disk (/s).

Memory
swpd : the amount of virtual memory used.
free: the amount of idle memory.
buff: the amount of memory used as buffers.
cache: the amount of memory used as cache.
inact: the amount of inactive memory. (-a option)
active: the amount of active memory. (-a option)

IO
bi: Blocks received from a block device (blocks/s).
bo: Blocks sent to a block device (blocks/s).

System
in: The number of interrupts per second, including the clock.
cs: The number of context switches per second.

CPU
us sy id wa st
These are percentages of total CPU time.
us: Time spent running non-kernel code. (user time, including nice time)
sy: Time spent running kernel code. (system time)
id: Time spent idle. Prior to Linux 2.5.41, this includes IO-wait time.
wa: Time spent waiting for IO. Prior to Linux 2.5.41, included in idle.
st: Time stolen from a virtual machine. Prior to Linux 2.6.11, unknown.
(Grey color implies deprecated).

>>

```
...
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0 158488 226084 457832 2353500    0    0  1136  1421 1159 4010 21  3 69  6  0
 1  0 158488 201120 457832 2354760    0    0   129  1808 1113 3572 26  2 71  1  0
 1  0 158488 187140 457844 2356244    0    0   125  2308  958 2474 24  2 72  2  0
 1  0 158488 165104 457944 2362996    0    0   375  1977  909 2476 25  1 72  2  0
 1  0 158488 204496 457956 2364656    0    0   129  1864 1000 3431 23  2 66  8  0
...
```

System fairly busy..

```
# vmstat --help

Usage:
 vmstat [options] [delay [count]]

Options:
 -a, --active           active/inactive memory
 -f, --forks            number of forks since boot
 -m, --slabs            slabinfo
 -n, --one-header       do not redisplay header
 -s, --stats            event counter statistics
 -d, --disk             disk statistics
 -D, --disk-sum         summarize disk statistics
 -p, --partition <dev>  partition specific statistics
 -S, --unit <char>      define display unit
 -w, --wide             wide output

 -h, --help     display this help and exit
 -V, --version  output version information and exit

For more details see vmstat(8).
#
```

dstat

<< From the man page >>

Dstat is a versatile replacement for vmstat, iostat and ifstat. Dstat overcomes some of the limitations and adds some extra features.

Dstat allows you to view all of your system resources instantly, you can eg. compare disk usage in combination with interrupts from your IDE controller, or compare the network bandwidth numbers directly with the disk throughput (in the same interval).

Dstat also cleverly gives you the most detailed information in columns and clearly indicates in what magnitude and unit the output is displayed. Less confusion, less mistakes, more efficient.

Dstat is unique in letting you aggregate block device throughput for a certain diskset or network bandwidth for a group of interfaces, ie. you can see the throughput for all the block devices that make up a single filesystem or storage system.

Dstat allows its data to be directly written to a CSV file to be imported and used by OpenOffice, Gnumeric or Excel to create graphs.

```
# dstat
You did not select any stats, using -cdngy by default.
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send|  in   out | int   csw
  3   1  95   1   0   0|  41k  201k|   0     0 | 119B 1163B| 508  3439
 18   3  79   0   0   0|   0     0 |1130B  210B|   0     0 | 705  1851
 24   3  73   0   0   0|   0     0 | 140B   70B|   0     0 | 755  1769
 23   3  73   1   0   0|   0   112k| 226B  140B|   0     0 | 770  1971
  1   1  87  10   0   0|   0   472k| 986B  955B|   0     0 | 654  2328
...
```

A very useful feature of dstat is the ability to quickly look up – at the granularity of a process – which process(es) is using the maximum resources.
A sample bash script function to achieve this:

```
dtop ()
{
    DLY=5;
    echo dstat --time --top-io-adv --top-cpu --top-mem ${DLY};
    dstat --time --top-io-adv --top-cpu --top-mem ${DLY}
}
...
```

```
$ dtop
dstat --time --top-io-adv --top-cpu --top-mem 5
----system---- -------most-expensive-i/o-process------- -most-expensive- --most-expensive-
     time     |process              pid  read write cpu| cpu process    |  memory process
```

```
24-12 12:15:09|dropbox              5898 2697k  31k1.2%|dropbox      1.2|dropbox     706M
24-12 12:15:14|chrome               3295 8811B9546B0.2%| --enable-cra0.6|dropbox     706M
24-12 12:15:19|chrome               3295 6208B4002B0.3%| --enable-cra0.6|dropbox     706M
24-12 12:15:24|chrome               3295 5952B3379B0.2%| --enable-cra0.7|dropbox     706M
24-12 12:15:26|chrome               3295   32k4226B0.4%|cinnamon     0.5|dropbox     706M
...
$
```

htop

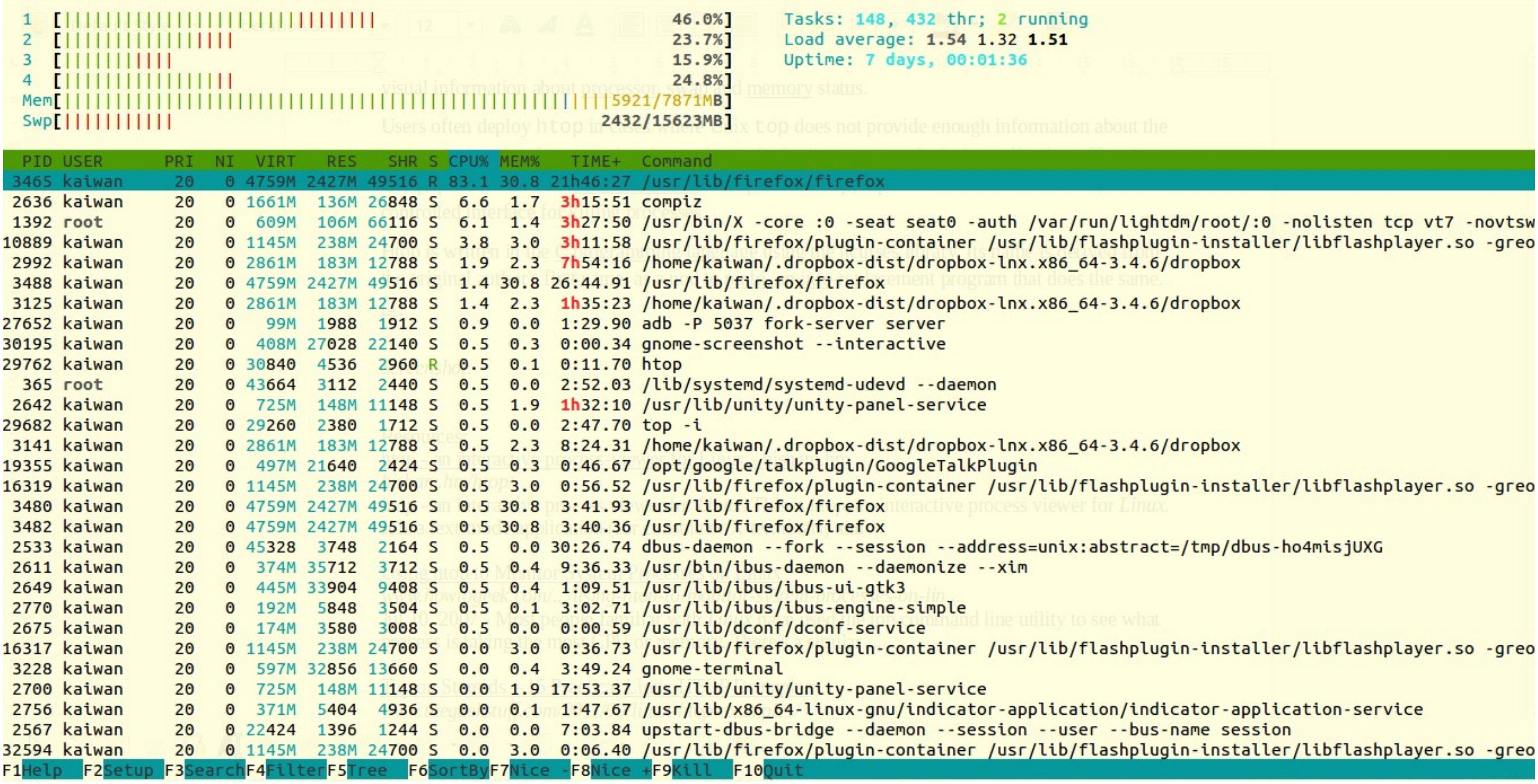
Htop is an interactive system-monitor process-viewer written for Linux. It is designed as an alternative to the Unix program top. It shows a frequently updated list of the processes running on a computer, normally ordered by the amount of CPU usage. Unlike `top`, htop provides a full list of processes running, instead of the top resource-consuming processes. Htop uses color and gives visual information about processor, swap and memory status.

Users often deploy `htop` in cases where Unix `top` does not provide enough information about the systems processes, for example when trying to find minor memory leaks in applications. Htop is also popularly used as a system monitor. Compared to `top`, it provides a more convenient, cursor-controlled interface for killing processes.

Htop is written in the C programming language using the ncurses library. Its name is derived from the original author's first name, as a nod to pinfo, an info-replacement program that does the same. [2]

P.T.O.

Screenshot:



```
  1  [||||||||||||||||||||||||||||||||||                  46.0%]   Tasks: 148, 432 thr; 2 running
  2  [|||||||||||||||||                                   23.7%]   Load average: 1.54 1.32 1.51
  3  [||||||||||                                          15.9%]   Uptime: 7 days, 00:01:36
  4  [||||||||||||||||                                    24.8%]
  Mem[|||||||||||||||||||||||||||||||||||||||||||||||||5921/7871MB]
  Swp[|||||||||||                                   2432/15623MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 3465 kaiwan     20   0 4759M 2427M 49516 R 83.1 30.8 21h46:27 /usr/lib/firefox/firefox
 2636 kaiwan     20   0 1661M  136M 26848 S  6.6  1.7  3h15:51 compiz
 1392 root       20   0  609M  106M 66116 S  6.1  1.4  3h27:50 /usr/bin/X -core :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtsw
10889 kaiwan     20   0 1145M  238M 24700 S  3.8  3.0  3h11:58 /usr/lib/firefox/plugin-container /usr/lib/flashplugin-installer/libflashplayer.so -greo
 2992 kaiwan     20   0 2861M  183M 12788 S  1.9  2.3  7h54:16 /home/kaiwan/.dropbox-dist/dropbox-lnx.x86_64-3.4.6/dropbox
 3488 kaiwan     20   0 4759M 2427M 49516 S  1.4 30.8 26:44.91 /usr/lib/firefox/firefox
 3125 kaiwan     20   0 2861M  183M 12788 S  1.4  2.3  1h35:23 /home/kaiwan/.dropbox-dist/dropbox-lnx.x86_64-3.4.6/dropbox
27652 kaiwan     20   0   99M  1988  1912 S  0.9  0.0  1:29.90 adb -P 5037 fork-server server
30195 kaiwan     20   0  408M 27028 22140 S  0.5  0.3  0:00.34 gnome-screenshot --interactive
29762 kaiwan     20   0 30840  4536  2960 R  0.5  0.1  0:11.70 htop
  365 root       20   0 43664  3112  2440 S  0.5  0.0  2:52.03 /lib/systemd/systemd-udevd --daemon
 2642 kaiwan     20   0  725M  148M 11148 S  0.5  1.9  1h32:10 /usr/lib/unity/unity-panel-service
29682 kaiwan     20   0 29260  2380  1712 S  0.5  0.0  2:47.70 top -i
 3141 kaiwan     20   0 2861M  183M 12788 S  0.5  2.3  8:24.31 /home/kaiwan/.dropbox-dist/dropbox-lnx.x86_64-3.4.6/dropbox
19355 kaiwan     20   0  497M 21640  2424 S  0.5  0.3  0:46.67 /opt/google/talkplugin/GoogleTalkPlugin
16319 kaiwan     20   0 1145M  238M 24700 S  0.5  3.0  0:56.52 /usr/lib/firefox/plugin-container /usr/lib/flashplugin-installer/libflashplayer.so -greo
 3480 kaiwan     20   0 4759M 2427M 49516 S  0.5 30.8  3:41.93 /usr/lib/firefox/firefox
 3482 kaiwan     20   0 4759M 2427M 49516 S  0.5 30.8  3:40.36 /usr/lib/firefox/firefox
 2533 kaiwan     20   0 45328  3748  2164 S  0.5  0.0 30:26.74 dbus-daemon --fork --session --address=unix:abstract=/tmp/dbus-ho4misjUXG
 2611 kaiwan     20   0  374M 35712  3712 S  0.5  0.4  9:36.33 /usr/bin/ibus-daemon --daemonize --xim
 2649 kaiwan     20   0  445M 33904  9408 S  0.5  0.4  1:09.51 /usr/lib/ibus/ibus-ui-gtk3
 2770 kaiwan     20   0  192M  5848  3504 S  0.5  0.1  3:02.71 /usr/lib/ibus/ibus-engine-simple
 2675 kaiwan     20   0  174M  3580  3008 S  0.5  0.0  0:00.59 /usr/lib/dconf/dconf-service
16317 kaiwan     20   0 1145M  238M 24700 S  0.0  3.0  0:36.73 /usr/lib/firefox/plugin-container /usr/lib/flashplugin-installer/libflashplayer.so -greo
 3228 kaiwan     20   0  597M 32856 13660 S  0.0  0.4  3:49.24 gnome-terminal
 2700 kaiwan     20   0  725M  148M 11148 S  0.0  1.9 17:53.37 /usr/lib/unity/unity-panel-service
 2756 kaiwan     20   0  371M  5404  4936 S  0.0  0.1  1:47.67 /usr/lib/x86_64-linux-gnu/indicator-application/indicator-application-service
 2567 kaiwan     20   0 22424  1396  1244 S  0.0  0.0  7:03.84 upstart-dbus-bridge --daemon --session --user --bus-name session
32594 kaiwan     20   0 1145M  238M 24700 S  0.0  3.0  0:06.40 /usr/lib/firefox/plugin-container /usr/lib/flashplugin-installer/libflashplayer.so -greo
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Resources:

[htop - an interactive process-viewer for Linux - hisham.hm]
hisham.hm/htop/
htop - an interactive process viewer for *Linux*. This is *htop*, an interactive process viewer for *Linux*. It is a text-mode application (for console or X terminals) and ...

[Using htop to Monitor System Processes on Linux]
www.howtogeek.com/.../using-htop-to-monitor-system-processes-on-lin...
Jul 10, 2007 - Most people familiar with *Linux* have used the top command line utility to see what process is taking the most CPU or memory. There's a similar ...

[Top on Steroids – 15 Practical Linux HTOP Examples]
www.thegeekstuff.com/2011/09/linux-htop-examples/
Sep 14, 2011 - To use *htop*, you need to install it first. Go to *htop* download page, and download the binaries that corresponds to your *Linux* distribution and ...

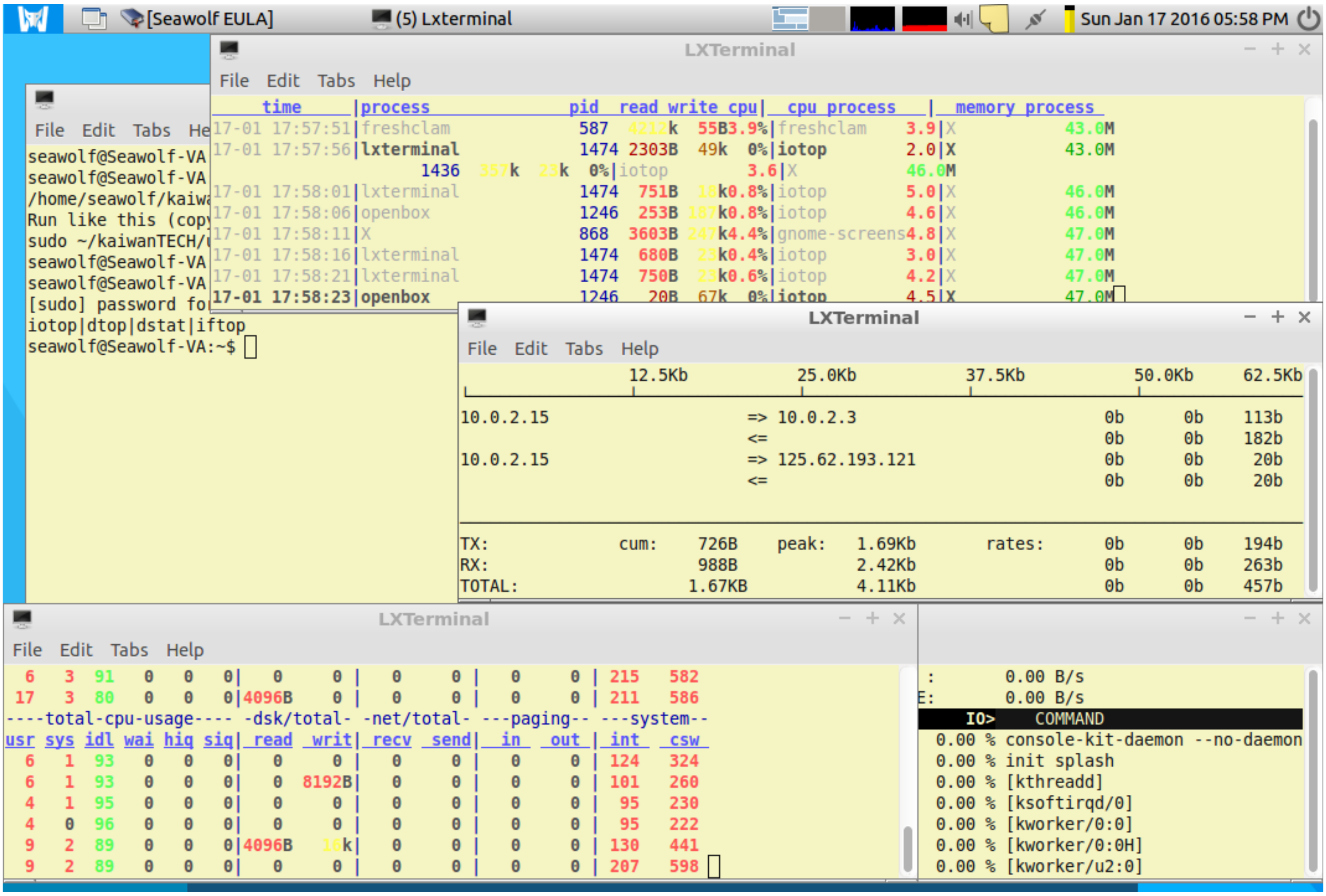
[System Monitoring with htop - Rackspace]
www.rackspace.com › ... › Cloud Hosting › Cloud Servers
Jun 12, 2014 - *htop* is a tool in *Linux* that allows you to monitor your system's vital ... *htop* is another tool to add in your system administrator toolkit and is ...

Simple System Monitoring for a Linux Desktop

Monday 06 January 2014 12:53 PM

The Problem- What exactly is eating into my HDD / processor / network right now?? Yeah! On the (Linux) desktop, we'd like to know why things crawl along sometimes. Which process(es) is the culprit behind that disk activity, or the memory hogger, or eating up network bandwidth? Many tools exist that can help us pinpoint … *Continue reading Simple System Monitoring for a Linux Desktop →*

Screenshot



```
seawolf@Seawolf-VA
seawolf@Seawolf-VA
/home/seawolf/kaiwa
Run like this (copy
sudo ~/kaiwanTECH/u
seawolf@Seawolf-VA
seawolf@Seawolf-VA
[sudo] password fo
iotop|dtop|dstat|iftop
seawolf@Seawolf-VA:~$ 
```

```
     time      |process                  pid  read write cpu|  cpu process    |  memory process
17-01 17:57:51|freshclam                587  4212k   55B3.9%|freshclam     3.9|X             43.0M
17-01 17:57:56|lxterminal              1474 2303B   49k   0%|iotop         2.0|X             43.0M
                         1436  357k   23k   0%|iotop         3.6|X             46.0M
17-01 17:58:01|lxterminal              1474  751B   18k0.8%|iotop         5.0|X             46.0M
17-01 17:58:06|openbox                 1246  253B  187k0.8%|iotop         4.6|X             46.0M
17-01 17:58:11|X                        868 3603B  247k4.4%|gnome-screens4.8|X             47.0M
17-01 17:58:16|lxterminal              1474  680B   23k0.4%|iotop         3.0|X             47.0M
17-01 17:58:21|lxterminal              1474  750B   23k0.6%|iotop         4.2|X             47.0M
17-01 17:58:23|openbox                 1246   20B   67k   0%|iotop         4.5|X             47.0M
```

```
            12.5Kb          25.0Kb          37.5Kb          50.0Kb     62.5Kb
└───────────────┴───────────────┴───────────────┴───────────────┴──────────
10.0.2.15                     => 10.0.2.3                       0b      0b     113b
                              <=                                0b      0b     182b
10.0.2.15                     => 125.62.193.121                 0b      0b      20b
                              <=                                0b      0b      20b

──────────────────────────────────────────────────────────────────────────────
TX:             cum:     726B     peak:   1.69Kb      rates:    0b      0b     194b
RX:                      988B             2.42Kb                0b      0b     263b
TOTAL:                  1.67KB            4.11Kb                0b      0b     457b
```

```
  6   3  91   0   0   0|   0     0 |   0     0 |   0     0 | 215   582
 17   3  80   0   0   0|4096B    0 |   0     0 |   0     0 | 211   586
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send|  in   out | int   csw
  6   1  93   0   0   0|   0     0 |   0     0 |   0     0 | 124   324
  6   1  93   0   0   0|   0  8192B|   0     0 |   0     0 | 101   260
  4   1  95   0   0   0|   0     0 |   0     0 |   0     0 |  95   230
  4   0  96   0   0   0|   0     0 |   0     0 |   0     0 |  95   222
  9   2  89   0   0   0|4096B   16k|   0     0 |   0     0 | 130   441
  9   2  89   0   0   0|   0     0 |   0     0 |   0     0 | 207   598
```

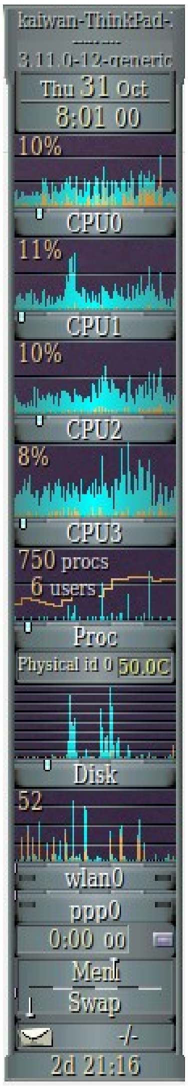
```
:        0.00 B/s
E:       0.00 B/s
    IO>    COMMAND
 0.00 % console-kit-daemon --no-daemon
 0.00 % init splash
 0.00 % [kthreadd]
 0.00 % [ksoftirqd/0]
 0.00 % [kworker/0:0]
 0.00 % [kworker/0:0H]
 0.00 % [kworker/u2:0]
```

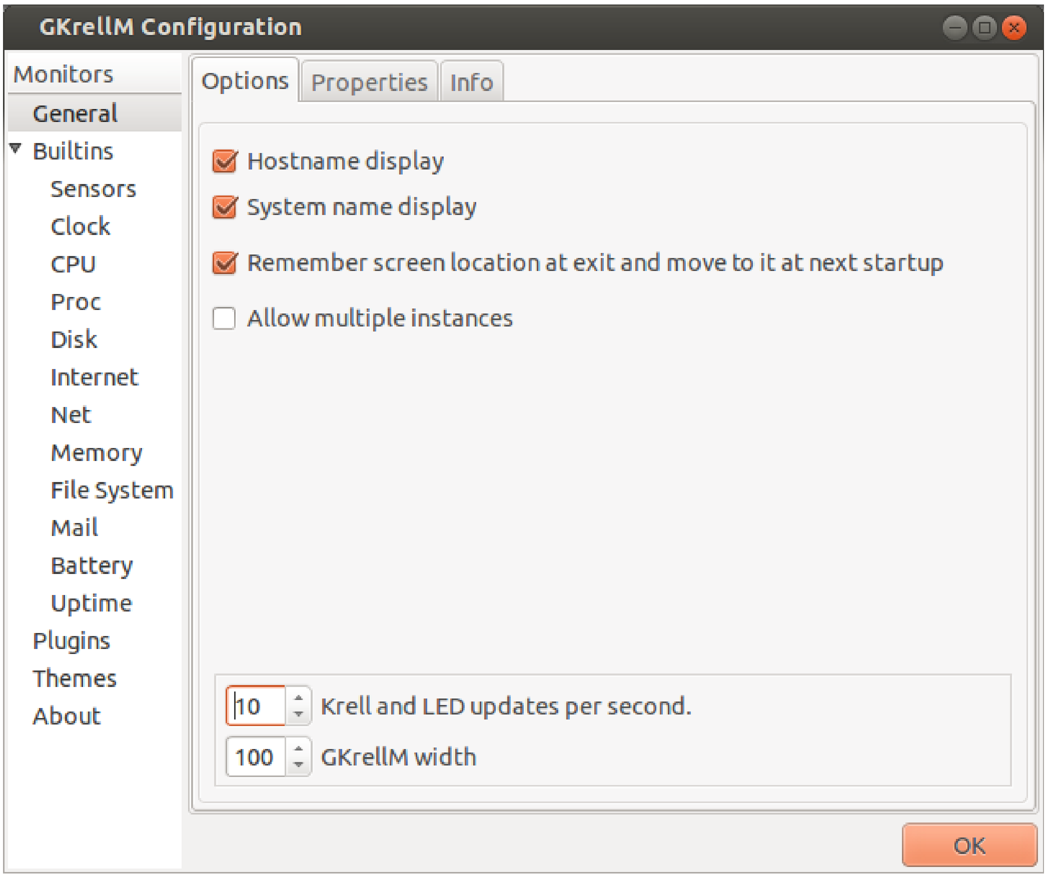
Networking

Can use **iftop** to monitor networking usage.

Packet capture and analysis: WireShark.
Also: tcpdump, netsniff-ng, ngrep.

GkrellM - Graphical GUI tool for system monitoring

GKrellM System Monitor Monitor for CPU, memory, disks, network, mail With a single process, gkrellm manages multiple stacked monitors and supports applying themes to match the monitors appearance to your window manager, Gtk, or any other theme. Developer website	



☑ Hostname display

☑ System name display

☑ Remember screen location at exit and move to it at next startup

□ Allow multiple instances

10 Krell and LED updates per second.

100 GKrellM width

nmon

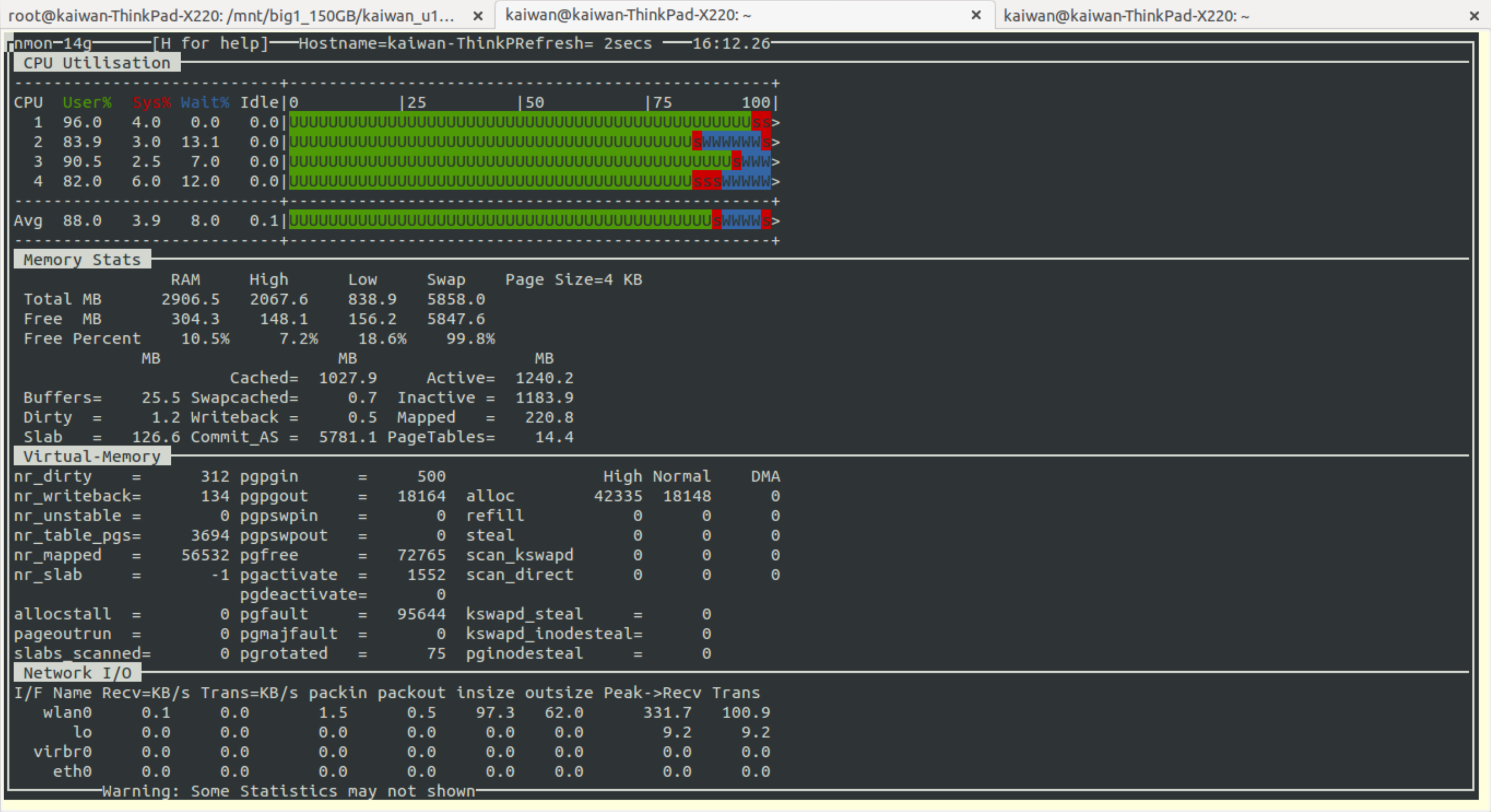
nmon is is a systems administrator, tuner, benchmark tool. It can display the CPU, memory, network, disks (mini graphs or numbers), file systems, NFS, top processes, resources (Linux version & processors) and on Power micro-partition information.

See the man page on nmon.
More than that:

```
$ nmon -h

Hint: nmon [-h] [-s <seconds>] [-c <count>] [-f -d <disks> -t -r <name>] [-x]

      -h            FULL help information
      Interactive-Mode:
      read startup banner and type: "h" once it is running
      For Data-Collect-Mode (-f)
      -f            spreadsheet output format [note: default -s300 -c288]
      optional
      -s <seconds>  between refreshing the screen [default 2]
      -c <number>   of refreshes [default millions]
      -d <disks>    to increase the number of disks [default 256]
      -t            spreadsheet includes top processes
      -x            capacity planning (15 min for 1 day = -fdt -s 900 -c 96)

Version - nmon 14g

For Interactive-Mode
      -s <seconds>  time between refreshing the screen [default 2]
      -c <number>   of refreshes [default millions]
      -g <filename> User Defined Disk Groups [hit g to show them]
                    - file = on each line: group_name <disks list> space
separated
                    - like: database sdb sdc sdd sde
                    - upto 64 disk groups, 512 disks per line
                    - disks can appear more than once and in many groups
      -b            black and white [default is colour]
      example: nmon -s 1 -c 100

For Data-Collect-Mode = spreadsheet format (comma separated values)
      Note: use only one of f,F,z,x or X and make it the first argument
      -f            spreadsheet output format [note: default -s300 -c288]
                   output file is <hostname>_YYYYMMDD_HHMM.nmon
      -F <filename> same as -f but user supplied filename
      -r <runname>  used in the spreadsheet file [default hostname]
      -t            include top processes in the output
      -T            as -t plus saves command line arguments in UARG section
      -s <seconds>  between snap shots
      -c <number>   of snapshots before nmon stops
      -d <disks>    to increase the number of disks [default 256]
      -l <dpl>      disks/line default 150 to avoid spreadsheet issues. EMC=64.
      -g <filename> User Defined Disk Groups (see above) - see BBBG & DG lines
      -N            include NFS Network File System
      -I <percent>  Include process & disks busy threshold (default 0.1)
```

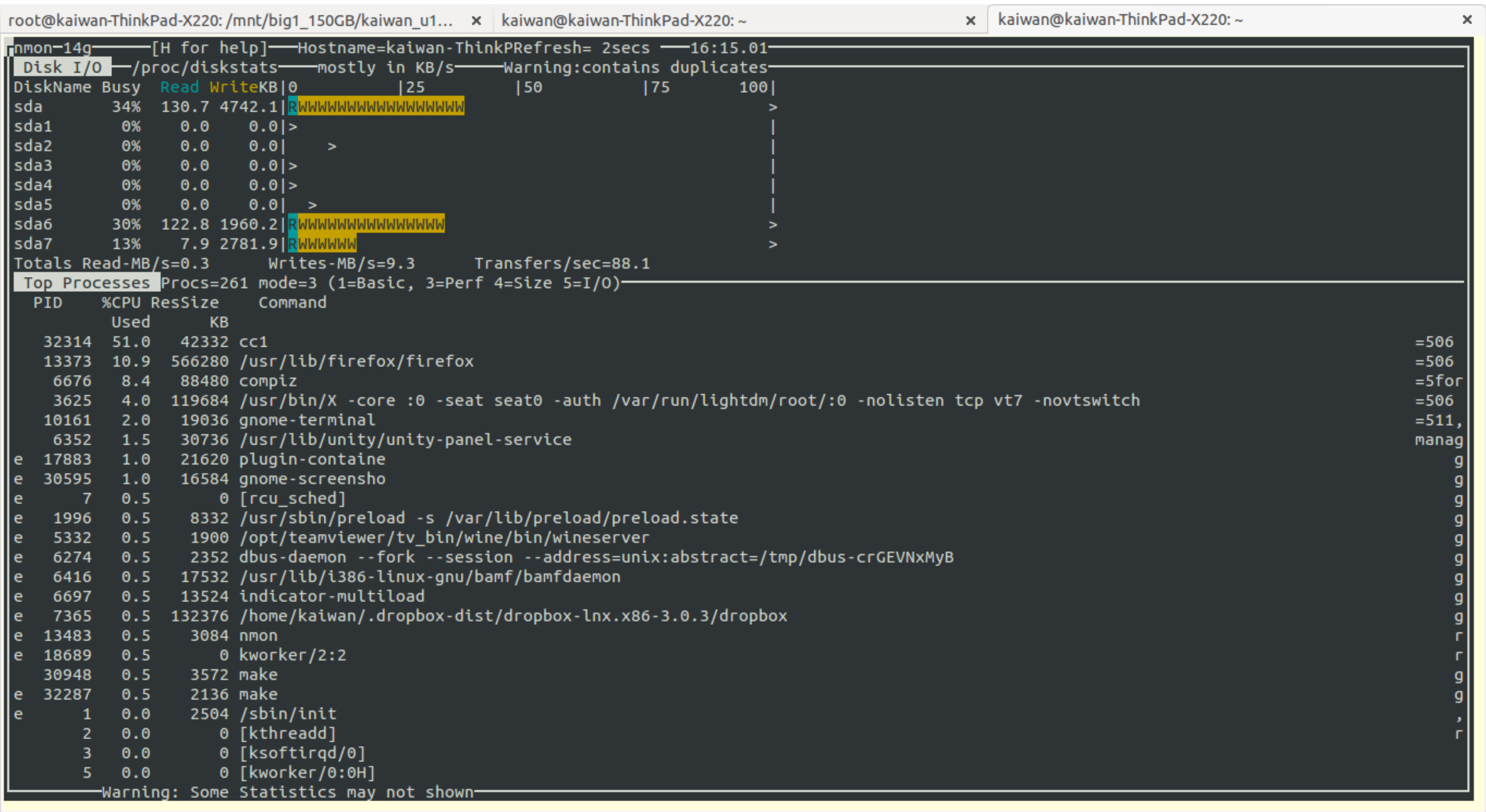
```
                    don't save or show proc/disk using less than this percent
      -m <directory> nmon changes to this directory before saving to file
      example: collect for 1 hour at 30 second intervals with top procs
             nmon -f -t -r Test1 -s30 -c120

      To load into a spreadsheet:
      sort -A *nmon >stats.csv
      transfer the stats.csv file to your PC
      Start spreadsheet & then Open type=comma-separated-value ASCII file
       The nmon analyser or consolidator does not need the file sorted.

Capacity planning mode - use cron to run each day
      -x            sensible spreadsheet output for CP =  one day
                    every 15 mins for 1 day ( i.e. -ft -s 900 -c 96)
      -X            sensible spreadsheet output for CP = busy hour
                    every 30 secs for 1 hour ( i.e. -ft -s 30 -c 120)

Interactive Mode Commands
      key --- Toggles to control what is displayed ---
      h   = Online help information
      r   = Machine type, machine name, cache details and OS version + LPAR
      c   = CPU by processor stats with bar graphs
      l   = long term CPU (over 75 snapshots) with bar graphs
      m   = Memory stats
      L   = Huge memory page stats
      V   = Virtual Memory and Swap stats
      k   = Kernel Internal stats
      n   = Network stats and errors
      N   = NFS Network File System
      d   = Disk I/O Graphs
      D   = Disk I/O Stats
      o   = Disk I/O Map (one character per disk showing how busy it is)
      o   = User Defined Disk Groups
      j   = File Systems
      t   = Top Process stats use 1,3,4,5 to select the data & order
      u   = Top Process full command details
      v   = Verbose mode - tries to make recommendations
      b   = black and white mode (or use -b option)
      .   = minimum mode i.e. only busy disks and processes

      key --- Other Controls ---
      +   = double the screen refresh time
      -   = halves the screen refresh time
      q   = quit (also x, e or control-C)
      0   = reset peak counts to zero (peak = ">")
      space = refresh screen now

Startup Control
      If you find you always type the same toggles every time you start
      then place them in the NMON shell variable. For example:
       export NMON=cmdrvtan

Others:
      a) To you want to stop nmon - kill -USR2 <nmon-pid>
      b) Use -p and nmon outputs the background process pid
      c) To limit the processes nmon lists (online and to a file)
         Either set NMONCMD0 to NMONCMD63 to the program names
         or use -C cmd:cmd:cmd etc. example: -C ksh:vi:syncd
      d) If you want to pipe nmon output to other commands use a FIFO:
         mkfifo /tmp/mypipe
         nmon -F /tmp/mypipe &
         grep /tmp/mypipe
      e) If nmon fails please report it with:
         1) nmon version like: 14g
         2) the output of cat /proc/cpuinfo
         3) some clue of what you were doing
         4) I may ask you to run the debug version

      Developer Nigel Griffiths
      Feedback welcome - on the current release only and state exactly the
problem
      No warranty given or implied.
$
```

```
$ export NMON=cmVn ; nmon
    <<
      c   = CPU by processor stats with bar graphs
      m   = Memory stats
      V   = Virtual Memory and Swap stats
      n   = Network stats and errors
    >>
```


root@kaiwan-ThinkPad-X220: /mnt/big1_150GB/kaiwan_u1...
kaiwan@kaiwan-ThinkPad-X220: ~
kaiwan@kaiwan-ThinkPad-X220: ~

```
nmon─14g─────────[H for help]───Hostname=kaiwan-ThinkPRefresh= 2secs ───16:12.26
 CPU Utilisation
---------------------------+-------------------------------------------------+
CPU  User%  Sys% Wait% Idle|0          |25         |50          |75       100|
  1  96.0   4.0   0.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUss>
  2  83.9   3.0  13.1   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUsWWWWWWs>
  3  90.5   2.5   7.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUsWWW>
  4  82.0   6.0  12.0   0.0|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUsssWWWWW>
---------------------------+-------------------------------------------------+
Avg  88.0   3.9   8.0   0.1|UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUsWWWWs>
---------------------------+-------------------------------------------------+
 Memory Stats
                RAM     High      Low     Swap    Page Size=4 KB
 Total MB     2906.5   2067.6    838.9   5858.0
 Free  MB      304.3    148.1    156.2   5847.6
 Free Percent   10.5%     7.2%    18.6%    99.8%
             MB                  MB                  MB
                     Cached=  1027.9     Active=  1240.2
 Buffers=    25.5 Swapcached=    0.7  Inactive =  1183.9
 Dirty  =     1.2 Writeback =    0.5  Mapped   =   220.8
 Slab   =   126.6 Commit_AS =  5781.1 PageTables=    14.4
 Virtual-Memory
nr_dirty    =       312 pgpgin      =     500               High Normal     DMA
nr_writeback=       134 pgpgout     =   18164  alloc       42335  18148       0
nr_unstable =         0 pgpswpin    =       0  refill          0      0       0
nr_table_pgs=      3694 pgpswpout   =       0  steal           0      0       0
nr_mapped   =     56532 pgfree      =   72765  scan_kswapd     0      0       0
nr_slab     =        -1 pgactivate  =    1552  scan_direct     0      0       0
                        pgdeactivate=       0
allocstall  =         0 pgfault     =   95644  kswapd_steal     =     0
pageoutrun  =         0 pgmajfault  =       0  kswapd_inodesteal=     0
slabs_scanned=        0 pgrotated   =      75  pginodesteal     =     0
 Network I/O
I/F Name Recv=KB/s Trans=KB/s packin packout insize outsize Peak->Recv Trans
   wlan0     0.1      0.0        1.5      0.5    97.3   62.0      331.7   100.9
      lo     0.0      0.0        0.0      0.0     0.0    0.0        9.2     9.2
  virbr0     0.0      0.0        0.0      0.0     0.0    0.0        0.0     0.0
    eth0     0.0      0.0        0.0      0.0     0.0    0.0        0.0     0.0
─────────Warning: Some Statistics may not shown─────────────────────────────────
```

```
$ export NMON=duv ; nmon
      <<
         d   = Disk I/O Graphs
         u   = Top Process full command details
        v   = Verbose mode - tries to make recommendations

       >>
```


root@kaiwan-ThinkPad-X220: /mnt/big1_150GB/kaiwan_u1…
kaiwan@kaiwan-ThinkPad-X220: ~
kaiwan@kaiwan-ThinkPad-X220: ~

```
┌nmon─14g──────[H for help]───Hostname=kaiwan-ThinkPRefresh= 2secs ───16:15.01─────────────────────────────────────────────────────────────
│ Disk I/O ──/proc/diskstats────mostly in KB/s─────Warning:contains duplicates───────────────────────────────────────────────────────────
│DiskName Busy  Read WriteKB|0          |25         |50          |75        100|
│sda        34%  130.7 4742.1|RWWWWWWWWWWWWWWWWWW                              >
│sda1        0%    0.0    0.0|>                                                |
│sda2        0%    0.0    0.0|    >                                            |
│sda3        0%    0.0    0.0|>                                                |
│sda4        0%    0.0    0.0|>                                                |
│sda5        0%    0.0    0.0|  >                                              |
│sda6       30%  122.8 1960.2|RWWWWWWWWWWWWWWWW                                >
│sda7       13%    7.9 2781.9|RWWWWWW                                          >
│Totals Read-MB/s=0.3      Writes-MB/s=9.3      Transfers/sec=88.1
│ Top Processes Procs=261 mode=3 (1=Basic, 3=Perf 4=Size 5=I/O)──────────────────────────────────────────────────────────────────────────
│  PID     %CPU ResSize    Command
│          Used       KB
│   32314  51.0   42332 cc1                                                                                                          =506
│   13373  10.9  566280 /usr/lib/firefox/firefox                                                                                     =506
│    6676   8.4   88480 compiz                                                                                                       =5for
│    3625   4.0  119684 /usr/bin/X -core :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch                  =506
│   10161   2.0   19036 gnome-terminal                                                                                               =511,
│    6352   1.5   30736 /usr/lib/unity/unity-panel-service                                                                           manag
│e  17883   1.0   21620 plugin-containe                                                                                                  g
│e  30595   1.0   16584 gnome-screensho                                                                                                  g
│e      7   0.5       0 [rcu_sched]                                                                                                      g
│e   1996   0.5    8332 /usr/sbin/preload -s /var/lib/preload/preload.state                                                                g
│e   5332   0.5    1900 /opt/teamviewer/tv_bin/wine/bin/wineserver                                                                         g
│e   6274   0.5    2352 dbus-daemon --fork --session --address=unix:abstract=/tmp/dbus-crGEVNxMyB                                          g
│e   6416   0.5   17532 /usr/lib/i386-linux-gnu/bamf/bamfdaemon                                                                            g
│e   6697   0.5   13524 indicator-multiload                                                                                                g
│e   7365   0.5  132376 /home/kaiwan/.dropbox-dist/dropbox-lnx.x86-3.0.3/dropbox                                                           g
│e  13483   0.5    3084 nmon                                                                                                             r
│e  18689   0.5       0 kworker/2:2                                                                                                      r
│   30948   0.5    3572 make                                                                                                             g
│e  32287   0.5    2136 make                                                                                                             g
│e      1   0.0    2504 /sbin/init                                                                                                         ,
│       2   0.0       0 [kthreadd]                                                                                                         r
│       3   0.0       0 [ksoftirqd/0]
│       5   0.0       0 [kworker/0:0H]
└─────────Warning: Some Statistics may not shown──────────────────────────────────────────────────────────────────────────────────────────
```

Using sar

http://perso.wanadoo.fr/sebastien.godard/
***sysstat* utilities** *(iostat, sar, sadf, mpstat, and sa). The sysstat utilities are a collection of performance-monitoring tools for Linux. These tools get their system information from the proc file system.*

sar: system activity reporter: a set of very powerful performance monitoring (perfmon) and bottleneck identification tools.

sar is like a wrapper around many existing system-gathering utilities.

“The **sysstat** package contains utilities to monitor system performance and usage activity. Sysstat contains various utilities, common to many commercial Unixes, and tools you can schedule via cron to collect and historize performance and activity data.

- iostat(1) reports CPU statistics and input/output statistics for devices, partitions and network filesystems.
- mpstat(1) reports individual or combined processor related statistics.
- pidstat(1) reports statistics for Linux tasks (processes) : I/O, CPU, memory, etc.
- sar(1) collects, reports and saves system activity information (CPU, memory, disks, interrupts, network interfaces, TTY, kernel tables,etc.)
- sadc(8) is the system activity data collector, used as a backend for sar.
- sa1(8) collects and stores binary data in the system activity daily data file. It is a front end to sadc designed to be run from cron.
- sa2(8) writes a summarized daily activity report. It is a front end to sar designed to be run from cron.
- sadf(1) displays data collected by sar in multiple formats (CSV, XML, etc.) This is useful to load performance data into a database, or import them in a spreadsheet to make graphs.
- sysstat(5) is just a manual page for sysstat configuration file, giving the meaning of environment variables used by sysstat commands.
- nfsiostat-sysstat(1) reports input/output statistics for network filesystems (NFS).
- cifsiostat(1) reports CIFS statistics.

Go to the Features page to display a list of sysstat's main features, and to the Matrix of activities to list all the possible activities for sar and corresponding options to use with sar and sadc.”

http://sysadmin.lk/howto-install-sar-ksar-system-activity-info-centos-redhat/

sar Tutorial

kSar: GUI front-end
https://grepora.com/2018/01/17/ksar-generating-graphs-from-sar-reports/

Collect and report Linux System Activity Information with sar

Fedora-specific:
https://computersecuritystudent.com/UNIX/FEDORA/lesson13/index.html

Useful:

"10 Useful Sar (Sysstat) Examples for UNIX / Linux Performance Monitoring
by RAMESH NATARAJAN "

Useful:
See the **Matrix of activities** – essentially, a table of which activities sar can keep track of (with the option switch mentioned).

One can always install the distributor version of sysstat, using the appropriate package manager. On Debian/Ubuntu/Mint:

```
sudo apt-get install sysstat
```

Below, we download and install by hand the latest stable version of sysstat.
Download

As of 13 February 2015, the latest stable version of sar is 11.0.3.

From: "10 Useful Sar (Sysstat) Examples for UNIX / Linux Performance Monitoring
by RAMESH NATARAJAN "

When manually configuring sar, make sure to use:

```
./configure --enable-install-cron
```

Note: Make sure to pass the option –enable-install-cron. This does the following automatically for you. If you don't configure sysstat with this option, you have to do this ugly job yourself manually.

```
# cat /etc/issue
Ubuntu 14.10 \n \l

# uname -a
Linux kaiwan-ThinkPad-X220 3.16.0-31-generic #43-Ubuntu SMP Tue Mar 10
17:37:36 UTC 2015 x86_64 x86_64 x86_64 GNU/Linux
# ls -l /usr/local/lib/sa/           << The sa[12] & sadc is
                                        installed under
/usr/local/lib/sa >>
total 92
-rwxr-xr-x 1 root root   980 Mar 13 18:26 sa1*
-rwxr-xr-x 1 root root  1391 Mar 13 18:26 sa2*
-rwxr-xr-x 1 root root 84040 Mar 13 18:26 sadc*
#
```

Collect the sar statistics using cron job – sa1 and sa2

Create sysstat file under */etc/cron.d* directory that will collect the historical sar data.

```
# vi /etc/cron.d/sysstat
*/10 * * * * root /usr/local/lib/sa/sa1 1 1
53 23 * * * root /usr/local/lib/sa/sa2 -A
#
```

If you’ve installed sysstat from source, the default location of sa1 and sa2 is /usr/local/lib/sa. If you’ve installed using your distribution update method (for example: yum, up2date, or apt-get), this might be /usr/lib/sa/sa1 and /usr/lib/sa/sa2.

<< *Older output below* >>

(On my Ubuntu 12.04 LTS box) sa1 and sa2 files are here:
/usr/lib/sysstat

Changes ENABLED="true" in */etc/default/sysstat* .

Cron entries:
/etc/cron.d/sysstat

Once the cron entry is enabled and running, use 'sar' to see stuff about system state.
It uses the (daily) logs generated here:

```
# ls -l /var/log/sysstat/
total 224K
-rw-r--r-- 1 root root 177K Aug  2 19:10 sa02
-rw-r--r-- 1 root root   36K Aug  3 10:22 sa03
#
```

(See url above for lots of examples)

Eg.

To see disk IO:

```
# sar -b
Linux 3.2.21 (kaiwan-ThinkPad-X220)   Thursday 02 August 2012    _i686_
     (4 CPU)

05:47:01  IST       tps      rtps      wtps   bread/s   bwrtn/s
05:48:01  IST      6.71      3.93      2.78     33.32    385.74
05:50:01  IST      3.48      1.05      2.43     31.46     47.38
05:55:01  IST      6.61      2.86      3.75    107.83     70.16
...
06:15:01  IST      4.03      1.42      2.62    318.56    125.16
06:16:01  IST      3.32      0.23      3.08      3.33    399.67
06:17:01  IST      2.68      0.05      2.63      0.80     49.58
Average:         6.05      3.01      3.04    104.66    102.56
#
```

<< the time interval to capture stats was initially set to 5 min; later made it 1 min >>

Eg to see memory used/freed/cached etc

```
# sar -r
Linux 3.2.21 (kaiwan-ThinkPad-X220)      Thursday 02 August 2012  _i686_   (4 CPU)

05:47:01  IST kbmemfree kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit  kbactive   kbinact
05:48:01  IST    215756   2757640     92.74     66376    710088   6015044     67.04   1800532    748684
05:50:01  IST    219772   2753624     92.61     66632    710552   5988884     66.75   1796152    749388
05:55:01  IST    189812   2783584     93.62     67432    738436   6003928     66.92   1808652    767520
...
06:17:01  IST    398764   2574632     86.59     53092    649132   5801668     64.66   1690800    685692
06:18:01  IST    401388   2572008     86.50     53252    645672   5795812     64.60   1690968    682312
06:19:01  IST    396728   2576668     86.66     53400    650764   5800700     64.65   1690992    687532
Average:       343353   2630043     88.45     53914    661357   5904121     65.81   1729417    700077
#
```

Eg. Use “sar -B” to generate paging statistics. i.e Number of KB paged in (and out) from disk per second.

```
# sar -B
Linux 3.2.21 (kaiwan-ThinkPad-X220)      Thursday 02 August 2012  _i686_   (4 CPU)

05:47:01  IST  pgpgin/s pgpgout/s   fault/s  majflt/s  pgfree/s pgscank/s pgscand/s pgsteal/s    %vmeff
05:48:01  IST     16.66    192.87    154.03      0.07    383.17      0.00      0.00      0.00      0.00
05:50:01  IST     15.73     23.69    186.56      0.23    299.08      0.00      0.00      0.00      0.00
05:55:01  IST     53.92     35.08   1057.40      0.48   1205.52      0.00      0.00      0.00      0.00
...
06:19:01  IST      3.13     27.46    545.74      0.02    692.15      0.00      0.00      0.00      0.00
06:20:01  IST      1.33     25.72    276.02      0.02    330.16      0.00      0.00      0.00      0.00
06:21:01  IST      0.33     25.93    547.96      0.02    672.74      0.00      0.00      0.00      0.00
Average:        46.32     48.25    608.41      0.45    787.93     27.55      0.00     20.69     75.11
#
```

etc etc!

iostat

Display :
-c   Display the CPU utilization report.
-d n Display the device utilization report., every n seconds
-x   Display extended statistics.
-z   Tell iostat to omit output for any devices for which there was no activity during the sample period.

```
$ iostat -c -d 3 -x -z
Linux 3.13.0-24-generic (kaiwan-desktop) Wednesday 24 December 2014
_x86_64_    (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          17.53    0.00    4.09    5.84    0.00   72.54
```

<< From 'man iostat':
CPU Utilization Report

The first report generated by the iostat command is the CPU Utilization Report. For multiprocessor systems, the CPU values are global averages among all processors. The report has the following format:

Show the percentage of
%user : % CPU utilization that occurred while executing at the user level (application).
%nice : % CPU utilization that occurred while executing at the user level with nice priority.
%system : % CPU utilization that occurred while executing at the system level (kernel).
%iowait : % time that the CPU or CPUs were idle during which the system had an outstanding disk I/O request.
%steal : % time spent in involuntary wait by the virtual CPU or CPUs while the hypervisor was servicing another virtual processor.
%idle : % time that the CPU or CPUs were idle and the system did not have an outstanding disk I/O request.
>>

```
Device: rrqm/s  wrqm/s  r/s  w/s  rkB/s   wkB/s  avgrq-sz avgqu-sz  await r_await
w_await svctm  %util
sda       0.00   18.00 115.33 7.67 1100.00 229.33   21.62   0.45     3.69    3.78    2.26
1.99     24.53
```

<< From 'man iostat':
Device: This column gives the device (or partition) name as listed in the /dev directory.
...
rrqm/s : The number of read requests merged per second that were queued to the device.
wrqm/s : The number of write requests merged per second that were queued to the device.
r/s : The number (after merges) of read requests completed per second for the device.
w/s : The number (after merges) of write requests completed per second for the device.
rsec/s (rkB/s, rMB/s) : The number of sectors (kilobytes, megabytes) read from the device per second.
wsec/s (wkB/s, wMB/s) : The number of sectors (kilobytes, megabytes) written to the device per second.
avgrq-sz : The average size (in sectors) of the requests that were issued to the device.
avgqu-sz : The average queue length of the requests that were issued to the device.
await : The average time (in milliseconds) for I/O requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.
r_await : The average time (in milliseconds) for read requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.
w_await : The average time (in milliseconds) for write requests issued to the device to be served. This includes the time spent by the requests in queue and the time spent servicing them.
svctm : The average service time (in milliseconds) for I/O requests that were issued to the device. Warning! Do not trust this field any more. This field will be removed in a future sysstat version.

%util : *Percentage of CPU time during which I/O requests were issued to the device (bandwidth utilization for the device). Device saturation occurs when this value is close to 100%.*

```
>>
...
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          25.17    0.00    1.83    0.58    0.00   72.42

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz
await r_await w_await  svctm  %util
sda               0.00     8.67    4.00    6.00    73.33  1449.33   304.53     0.02
2.13    0.00    3.56   2.00   2.00

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          24.96    0.00    2.25    0.83    0.00   71.95

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz
await r_await w_await  svctm  %util
sda               0.00     5.00    4.67    5.67   110.67  1470.67   306.06     0.02
1.68    0.29    2.82   1.29   1.33

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          25.88    0.00    2.34    0.83    0.00   70.95

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz
await r_await w_await  svctm  %util
sda               0.00     5.00    4.67   18.33    41.33  1721.33   153.28     0.01
0.64    0.00    0.80   0.46   1.07

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           8.02    0.00    1.25    0.84    0.00   89.89

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz
await r_await w_await  svctm  %util
sda               0.00     7.00    1.67    3.00    13.33   401.33   177.71     0.03
5.43    3.20    6.67   5.43   2.53

...
#
```

Related: pidstat(1), mpstat(1), vmstat(8)

---

blktrace

See the man page, with examples.

Interpreting Raw Statistics from the Kernel

The Linux kernel stores disk accounting information and stats within both the /proc and /sys pseduo-filesystems.

Documentation: https://www.kernel.org/doc/Documentation/iostats.txt

```
I/O statistics fields
---------------

Since 2.4.20 (and some versions before, with patches), and 2.5.45,
more extensive disk statistics have been introduced to help measure disk
activity. Tools such as sar and iostat typically interpret these and do
the work for you, but in case you are interested in creating your own
tools, the fields are explained here.

In 2.4 now, the information is found as additional fields in
/proc/partitions.  In 2.6, the same information is found in two
places: one is in the file /proc/diskstats, and the other is within
the sysfs file system, which must be mounted in order to obtain
the information. Throughout this document we'll assume that sysfs
is mounted on /sys, although of course it may be mounted anywhere.
Both /proc/diskstats and sysfs use the same source for the information
and so should not differ.

Here are examples of these different formats:

2.4:
   3     0   39082680 hda 446216 784926 9550688 4382310 424847 312726 5922052
19310380 0 3376340 23705160
   3     1    9221278 hda1 35486 0 35496 38030 0 0 0 0 0 38030 38030


2.6 sysfs:
   446216 784926 9550688 4382310 424847 312726 5922052 19310380 0 3376340
23705160
   35486    38030    38030    38030

2.6 diskstats:
   3    0   hda 446216 784926 9550688 4382310 424847 312726 5922052 19310380 0
3376340 23705160
   3    1   hda1 35486 38030 38030 38030

On 2.4 you might execute "grep 'hda ' /proc/partitions". On 2.6, you have
a choice of "cat /sys/block/hda/stat" or "grep 'hda ' /proc/diskstats".
The advantage of one over the other is that the sysfs choice works well
if you are watching a known, small set of disks.  /proc/diskstats may
be a better choice if you are watching a large number of disks because
you'll avoid the overhead of 50, 100, or 500 or more opens/closes with
each snapshot of your disk statistics.

In 2.4, the statistics fields are those after the device name. In
the above example, the first field of statistics would be 446216.
```

By contrast, in 2.6 if you look at /sys/block/hda/stat, you'll find just the eleven fields, beginning with 446216. If you look at /proc/diskstats, the eleven fields will be preceded by the major and minor device numbers, and device name.

Each of these formats provides eleven fields of statistics, each meaning exactly the same things. All fields except field 9 are cumulative since boot. Field 9 should go to zero as I/Os complete; all others only increase (unless they overflow and wrap). Yes, these are (32-bit or 64-bit) unsigned long (native word size) numbers, and on a very busy or long-lived system they may wrap. Applications should be prepared to deal with that; unless your observations are measured in large numbers of minutes or hours, they should not wrap twice before you notice them.

[P.T.O.]

```
Each set of stats only applies to the indicated device; if you want
system-wide stats you'll have to find all the devices and sum them all up.
```

Reads : fields 1-4.
Cumulative

```
Field  1 -- # of reads completed
    This is the total number of reads completed successfully.
Field  2 -- # of reads merged, field 6 -- # of writes merged
    Reads and writes which are adjacent to each other may be merged for
    efficiency.  Thus two 4K reads may become one 8K read before it is
    ultimately handed to the disk, and so it will be counted (and queued)
    as only one I/O.  This field lets you know how often this was done.
Field  3 -- # of sectors read
    This is the total number of sectors read successfully.
Field  4 -- # of milliseconds spent reading
    This is the total number of milliseconds spent by all reads (as
    measured from __make_request() to end_that_request_last()).
```

Writes : fields 5-8.
Cumulative

```
Field  5 -- # of writes completed
    This is the total number of writes completed successfully.
Field  6 -- # of writes merged
    See the description of field 2.
Field  7 -- # of sectors written
    This is the total number of sectors written successfully.
Field  8 -- # of milliseconds spent writing
    This is the total number of milliseconds spent by all writes (as
    measured from __make_request() to end_that_request_last()).



Field  9 -- # of I/Os currently in progress
    The only field that should go to zero. Incremented as requests are
    given to appropriate struct request_queue and decremented as they finish.


Field 10 -- # of milliseconds spent doing I/Os
    This field increases so long as field 9 is nonzero.
Field 11 -- weighted # of milliseconds spent doing I/Os
    This field is incremented at each I/O start, I/O completion, I/O
    merge, or read of these stats by the number of I/Os in progress
    (field 9) times the number of milliseconds spent doing I/O since the
    last update of this field.  This can provide an easy measure of both
    I/O completion time and the backlog that may be accumulating.
```

Reads completed *Reads merged* *Sectors read* *# ms reading* *# I/Os in progress* *# ms doing I/Os* *Weighted # ms doing I/Os*

```
<<
Eg.
# while [ true ] ; do  cat /sys/block/sda/sda7/stat ; sleep 1 ; done
    1          2       3        4          5          6       7        8           9     10       11
  913078     20345 27130450 18524332    542206    382325 17047792 70554076       0 9383032 89078332
  913091     20345 27130594 18524444    542206    382325 17047792 70554076       0 9383144 89078444
  913091     20345 27130594 18524444    542206    382325 17047792 70554076       0 9383144 89078444
  913091     20345 27130594 18524444    542206    382325 17047792 70554076       0 9383144 89078444
  913099     20345 27130698 18524580    542206    382325 17047792 70554076       0 9383188 89078580
```

Field #

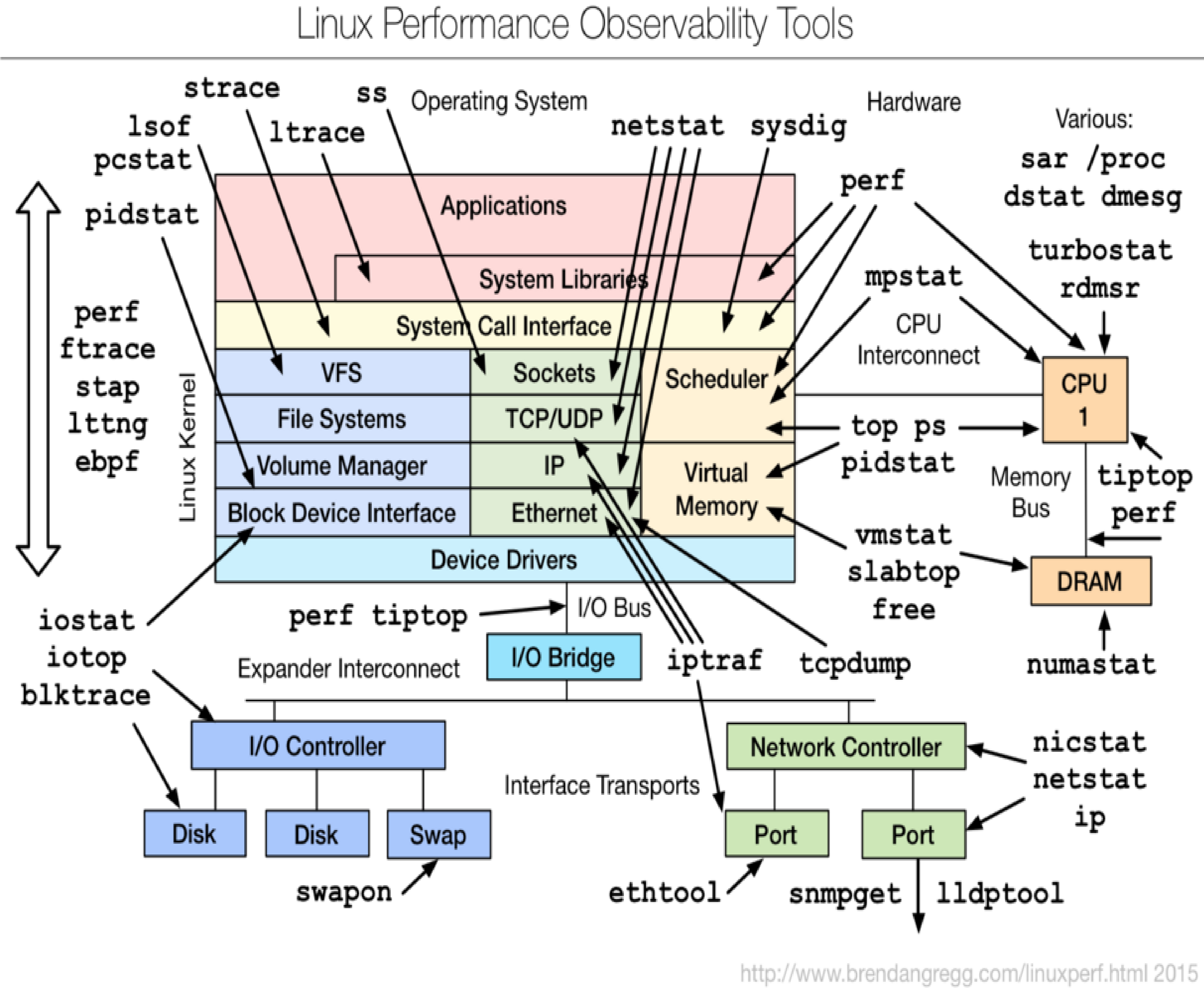
Writes completed *Writes merged* *Sectors written* *# ms writing*

```
--snip--
  913125     20345 27131570 18525080    542210    382335 17047904 70554108       0 9383476 89079112
  913144     20345 27132154 18525316    542213    382349 17048040 70554140      33 9383644 89079644
  913212     20345 27132698 18527052    542213    382349 17048040 70554140       0 9383732 89081116
```

```
--snip--
  913657    20345 27140066 18537292   542313   382483 17058968 70555996        0  9385604 89093208
  913661    20345 27140322 18537312   542313   382483 17058968 70555996       30  9385632 89093468
  913783    20345 27184866 18539980   542386   382485 17094968 70558292        2  9386636 89098200
  913888    20345 27237610 18541864   542445   382485 17152312 70561196        2  9387636 89103068
  914000    20345 27294442 18543664   542501   382485 17206328 70562856        6  9388640 89107144
  914114    20345 27351794 18545820   542558   382485 17260856 70564716        8  9389640 89110568
  914237    20347 27411954 18548380   542626   382515 17324600 70567356        2  9390828 89115668
  914316    20347 27447530 18550152   542682   382522 17360648 70570596        0  9391688 89120664
  914316    20347 27447530 18550152   542682   382522 17360648 70570596        0  9391688 89120664
^C
#
```

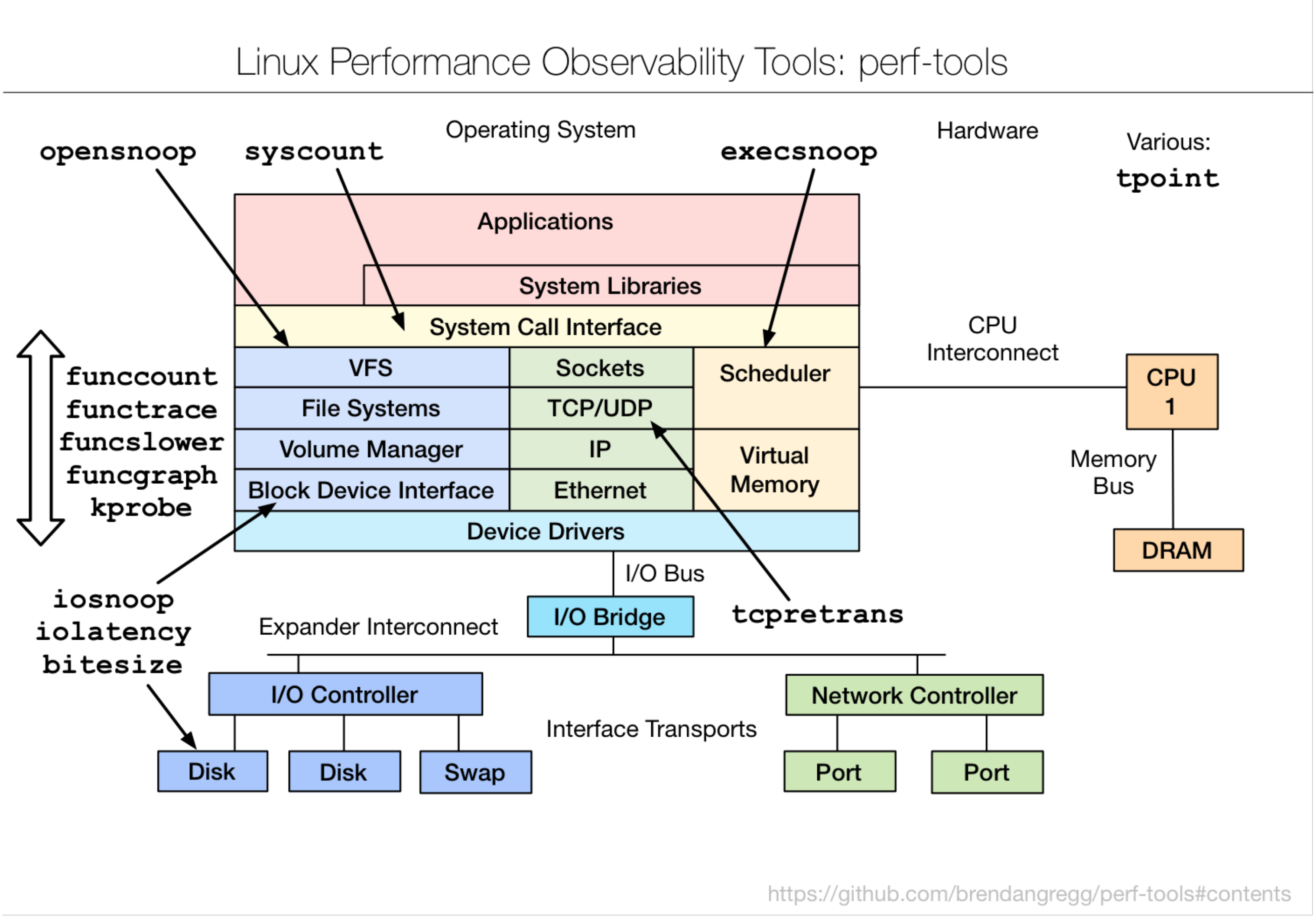
```
>>

To avoid introducing performance bottlenecks, no locks are held while
modifying these counters.  This implies that minor inaccuracies may be
introduced when changes collide, so (for instance) adding up all the
read I/Os issued per partition should equal those made to the disks ...
but due to the lack of locking it may only be very close.

--snip--

In 2.6, all disk statistics were removed from /proc/stat.  In 2.4, they
appear in both /proc/partitions and /proc/stat, although the ones in
/proc/stat take a very different format from those in /proc/partitions
(see proc(5), if your system has it.)

-- ricklind@us.ibm.com
```

Source
Linux Performance Observability Tools, Brendan Gregg



Source

Linux Performance Observability Tools: Perf Tools, Brendan Gregg



!WARNING! Many of these tools are considered to be Experimental and Unstable!

<< Optional / FYI >>

Network performance

Measure the quality and throughput of a network link using an excellent, open-source, scriptable & easy-to-use tool iPerf. A good Iperf tutorial is avialble here.

Source: Essential Linux Device Drivers, S Venkateshwaran

--*snip*--

Network Throughput

Several tools are available to benchmark network performance. Netperf, available for free from www.netperf.org, can set up complex TCP/UDP connection scenarios. You can use scripts to control characteristics such as protocol parameters, number of simultaneous sessions, and size of data blocks.

Benchmarking is accomplished by comparing the resulting throughput with the maximum practical bandwidth that the networking technology yields. For example, a 155Mbps ATM adapter produces a maximum IP throughput of 135Mbps, taking into account the ATM cell header size, overheads due to the ATM Adaptation Layer (AAL), and the occasional maintenance cells sent by the physical Synchronous Optical Networking (SONET) layer.

To obtain optimal throughput, you have to design your NIC driver for high performance. In addition, you need an in-depth understanding of the network protocol that your driver ferries.

...

FYI: Commands to spew out hardware-related information:

- `procfs`
- `sysfs`
- `lshw`
- `lscpu`
- `lspci`
- `lsusb`
- `dmidecode  [x86]`

Source

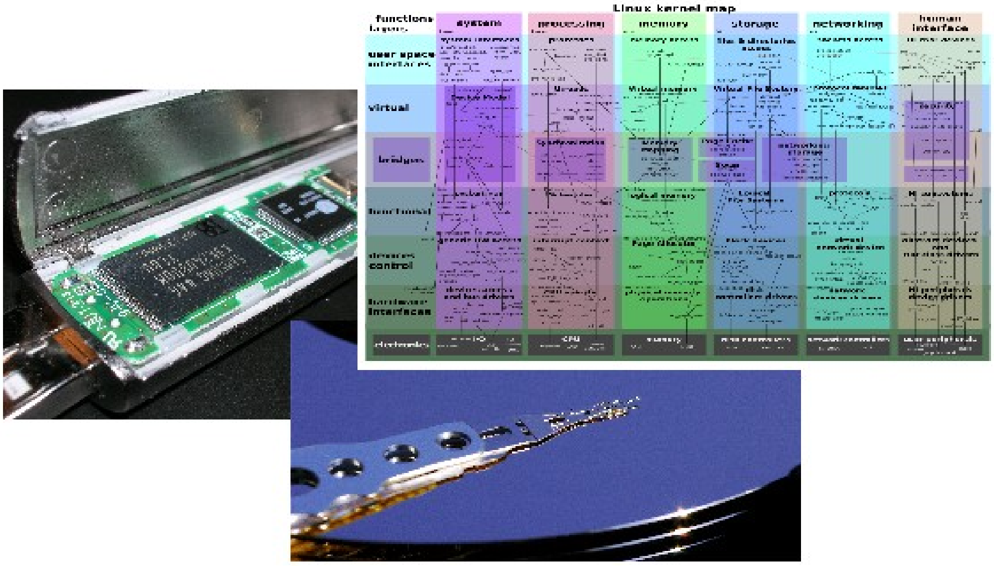
CPU Workload Characterization

Q. What's causing CPU load?

- Who?
- How?
- Why?
- What?

A.

Who?	:	top, htop
How?	:	system monitoring tools
Why?	:	perf record -g flame graphs
What?	:	perf stat -a -d



Linux kernel map
functions layers
system
processing
memory
storage
networking
human interface
user space interfaces
virtual
bridges
functional
devices control
hardware interfaces
electronics