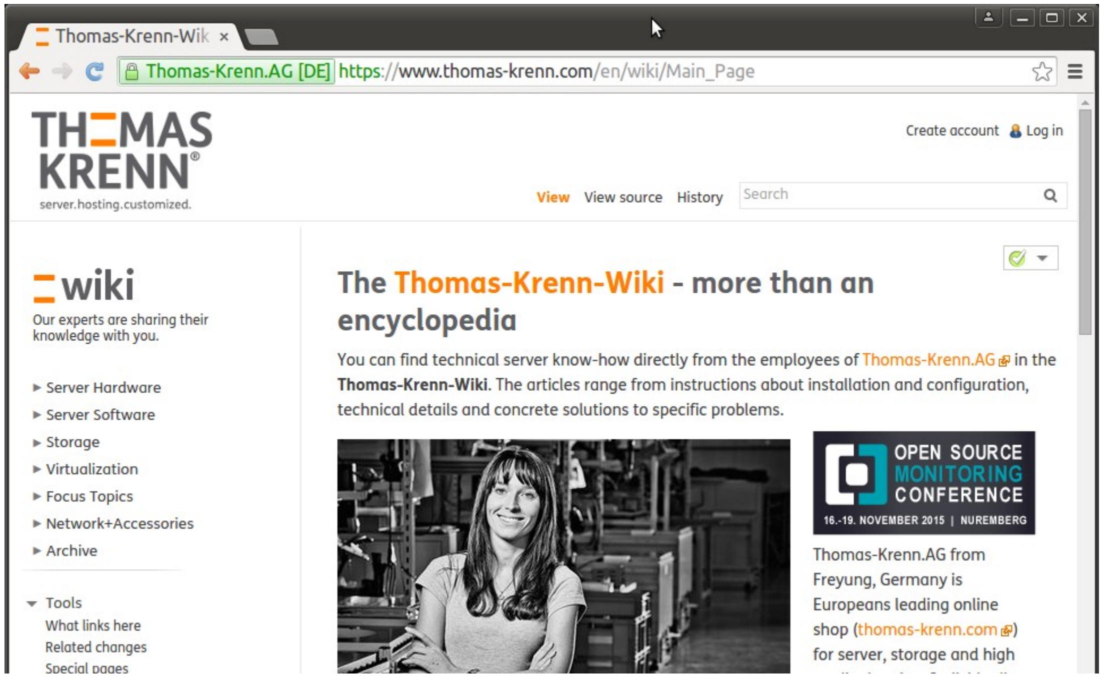
Linux Performance Profiling and Monitoring

Christoph Mitasch & Georg Schönberger
@cmitasch

THOMAS KRENN®
server.hosting.customized.

Thomas-Krenn.AG

- A server manufacturer in Bavaria, Germany
- Well visited knowledge base, Thomas-Krenn Wiki (parts in English)



Thomas-Krenn-Wiki

Thomas-Krenn.AG [DE] https://www.thomas-krenn.com/en/wiki/Main_Page

Create account Log in

View View source History Search

wiki
Our experts are sharing their knowledge with you.

- Server Hardware
- Server Software
- Storage
- Virtualization
- Focus Topics
- Network+Accessories
- Archive

Tools
What links here
Related changes
Special pages

The Thomas-Krenn-Wiki - more than an encyclopedia

You can find technical server know-how directly from the employees of Thomas-Krenn.AG in the Thomas-Krenn-Wiki. The articles range from instructions about installation and configuration, technical details and concrete solutions to specific problems.

OPEN SOURCE MONITORING CONFERENCE
16.-19. NOVEMBER 2015 | NUREMBERG

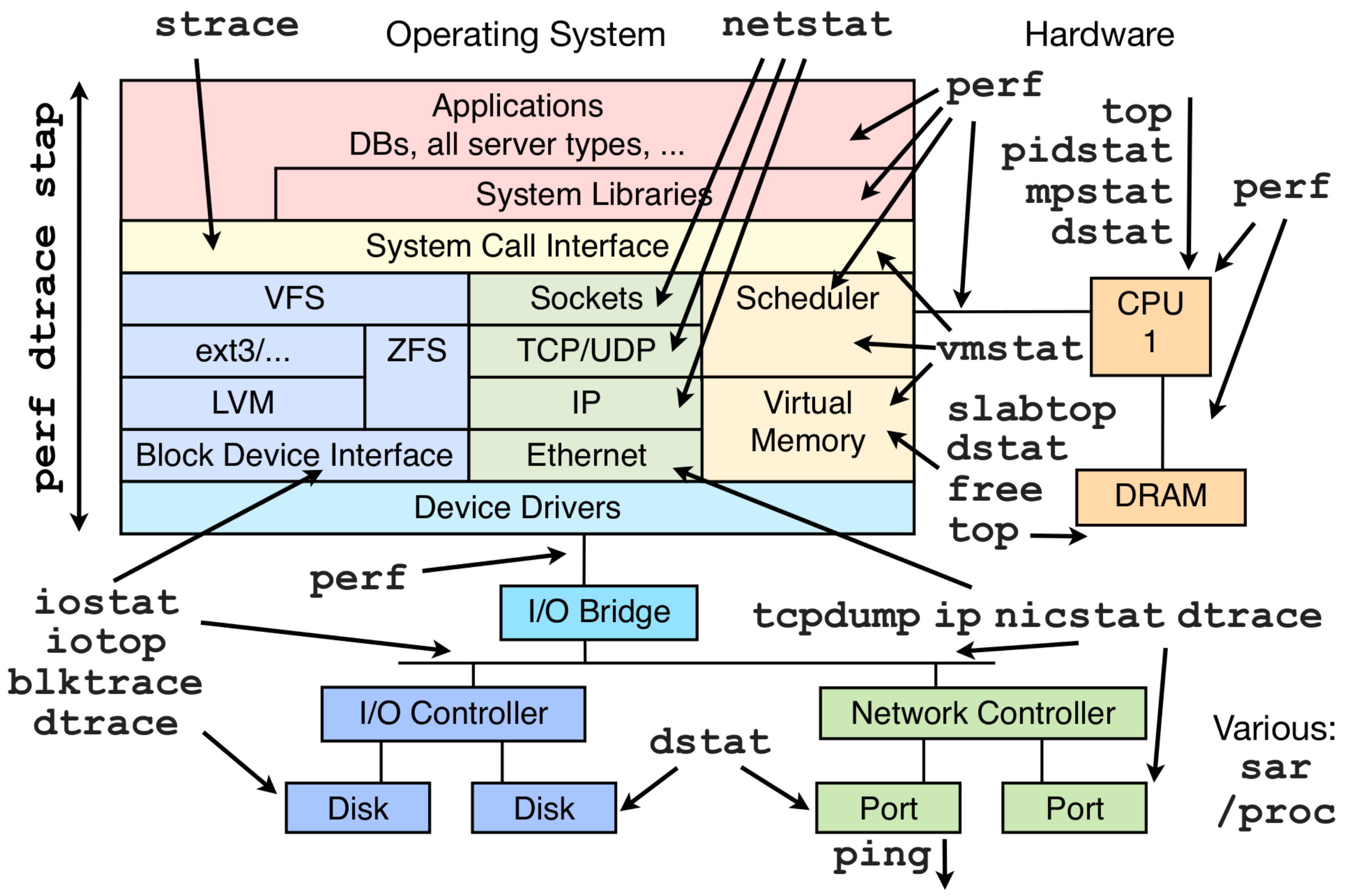
Thomas-Krenn.AG from Freyung, Germany is Europeans leading online shop (thomas-krenn.com) for server, storage and high

Agenda

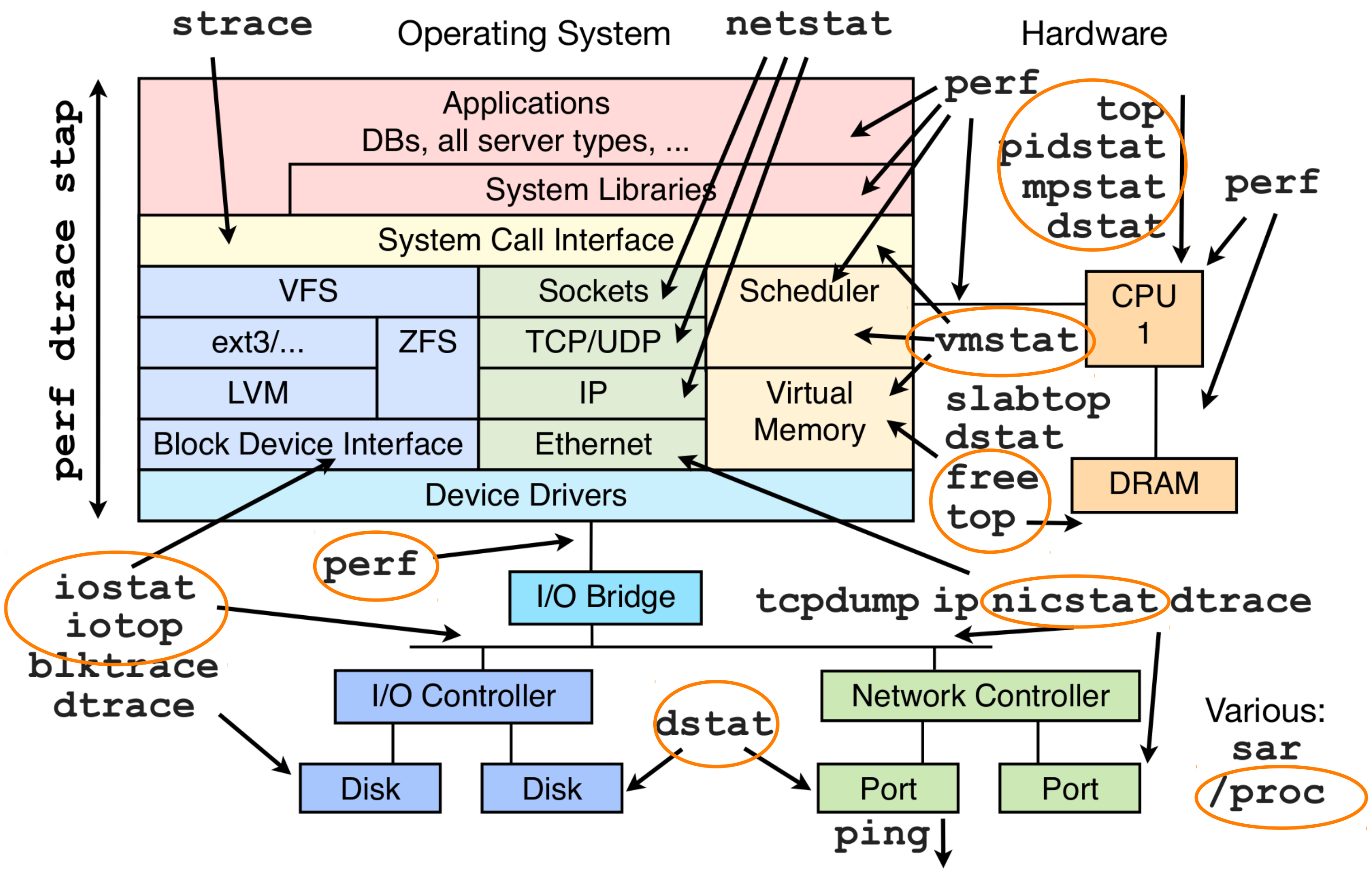
THOMAS KRENN® server.hosting.customized.

- Collect Statistics
  - Sysstat Package
    - iostat
    - pidstat
  - sar, atop
  - Percona Cacti Template
- Watch online
  - top
  - iotop
  - iftop
- Tracing
  - perf_events
  - ftrace
  - perf-tools
  - Flame graphs

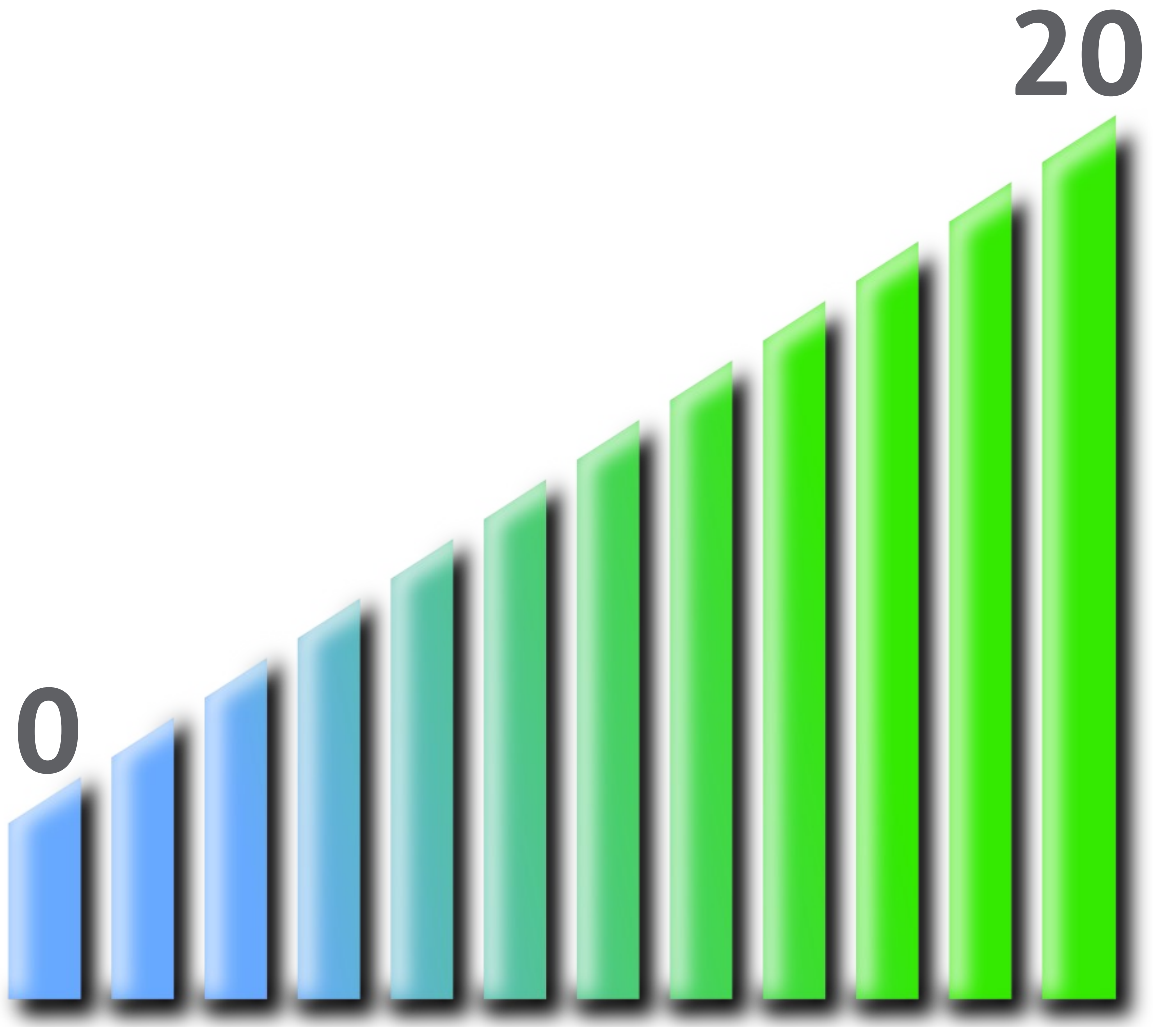
```
# find / -type f -name statistics
```



From Brendan Gregg, http://www.brendangregg.com/linuxperf.html



From Brendan Gregg, http://www.brendangregg.com/linuxperf.html


0
20

mpstat (part of sysstat)

- Without Interval/Count → since system startup
- CPU usage per Core
  - Including Hyperthreading

```
# lscpu | grep -E 'core|socket'
Thread(s) per core:    2
Core(s) per socket:    2
```

- Check how well usage is balanced

```
# mpstat -P ALL
Linux 3.13.0-48-generic (X220) 2015-04-14      _x86_64_   (4 CPU)
14:28:21     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
14:28:21     all   11,59    0,09    3,62    0,03    0,00    0,04    0,00    0,00    0,00   84,64
14:28:21       0    6,45    0,05    1,87    0,04    0,00    0,07    0,00    0,00    0,00   91,53
14:28:21       1   16,44    0,11    5,56    0,01    0,00    0,00    0,00    0,00    0,00   77,89
14:28:21       2   17,15    0,14    5,55    0,03    0,00    0,05    0,00    0,00    0,00   77,08
14:28:21       3   16,27    0,11    4,89    0,01    0,00    0,02    0,00    0,00    0,00   78,70
```

mpstat

```
# mpstat -P ALL 1 2
Linux 3.13.0-48-generic (X220) 2015-04-14      _x86_64_   (4 CPU)

15:24:44     CPU     %usr    %nice     %sys  %iowait     %irq    %soft   %steal   %guest   %gnice    %idle
15:24:45     all     5,21     0,00     7,12    17,81     0,00     0,27     0,00     0,00     0,00    69,59
15:24:45       0     1,43     0,00     1,43     0,00     0,00     2,86     0,00     0,00     0,00    94,29
15:24:45       1    11,88     0,00    23,76    64,36     0,00                       0,00     0,00     0,00
15:24:45       2     4,12     0,00     1,03     0,00     0,00                       0,00     0,00    94,85
15:24:45       3     3,03     0,00     1,01     0,00     0,00                       0,00     0,00    95,96

15:24:45     CPU     %usr    %nice     %sys  %iowait     %irq                      %guest   %gnice    %idle
15:24:46     all     5,74     0,00     7,10    17,76     0,00                       0,00     0,00    68,85
15:24:46       0     2,99     0,00     1,49     0,00     0,00     2,99     0,00     0,00     0,00    92,54
15:24:46       1    11,88     0,00    23,76    64,36     0,00     0,00     0,00     0,00     0,00     0,00
15:24:46       2     6,00     0,00     1,00     0,00     0,00     0,00     0,00     0,00     0,00    93,00
15:24:46       3     1,01     0,00     1,01     0,00     0,00     0,00     0,00     0,00     0,00    97,98
```

Core 1 is not idle and also deals with %iowait

vmstat

- High Level Statistics about
  - Virtual memory
  - Swap/Paging
  - I/O statistics
  - System interrupts and context switches
  - CPU statistics

```
# vmstat 1
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 3  0    172 371856 137088 3125664    0    0     0 153060 7618 7059 17  9 56 17  0
 3  0    172 416596 137096 3125704    0    0     0 163420 8689 7419 11 10 61 17  0
 0  0    172 451716 137096 3089916    0    0     0     0  396 1848  3  1 96  0  0
 0  0    172 413916 137108 3118796    0    0     0    52  502 2218  9  2 90  0  0
 2  0    172 399756 137108 3118860    0    0 284884     0 14830 10941 10 13 66 12  0
 1  1    172 364948 137108 3118988    0    0 310792     0 16204 12738 20 13 53 14  0
```

vmstat

_ Memory statistics

- `buff` Raw disk blocks like filesystem metadata (superblocks, inodes)
- `cache` Memory used for data -> pages with actual contents

```
$ vmstat 1
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    172 607760 182172 3313684    0    0   159   496  154  222 18  6 76  0  0
 0  0    172 607628 182172 3313684    0    0     0    52  387 2008  4  2 95  0  0
 0  0    172 607348 182172 3313684    0    0     0     0  397 2034  4  1 95  0  0
 0  0    172 606448 182172 3313684    0    0     0     0  378 1896  4  2 94  0  0
$ free
             total       used       free     shared    buffers     cached
Mem:       8056664    7450316     606348     491820     182172    3313684
-/+ buffers/cache:    3954460    4102204
Swap:      1048572        172    1048400
```

vmstat

_ Process related fields

- r The number of runnable processes (running or waiting for run time)
  - If high → indicator for saturation
- b The number of processes in uninterruptible sleep
  - Mostly waiting for I/O

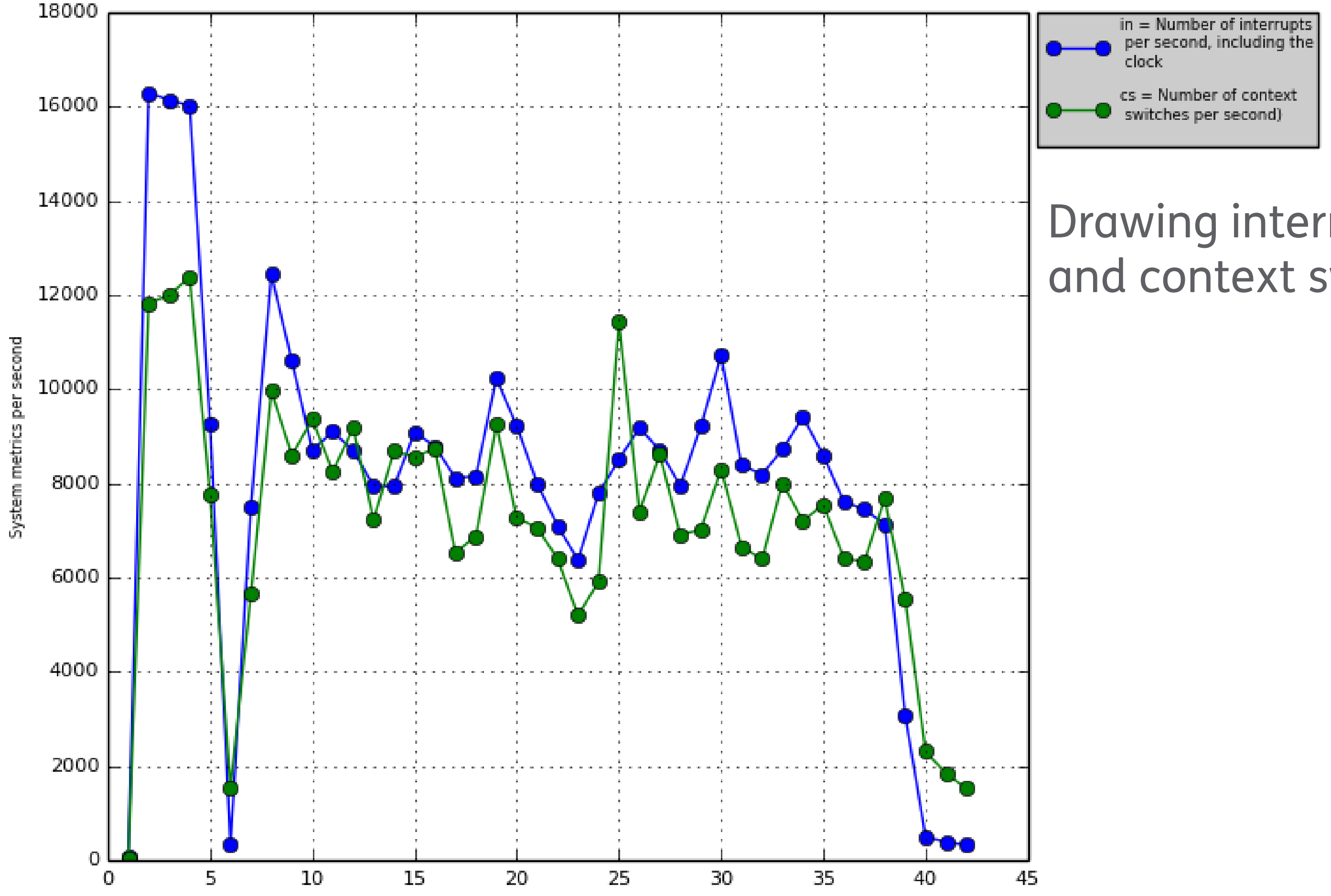
```
# vmstat 1
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd                                   bi    bo   in   cs us sy id wa st
[...]
 0  1    172 4                             0         167524 9029 6955  6  6 70 18  0
 0  1    172 3                             0      0 138340 8133 6165  7  7 68 19  0
$ ps -eo ppid,pid,user,stat,pcpu,comm,wchan:32 | grep ext4
[...]
 7159  7161 root     Ds    3.2 fio               ext4_file_write
 7159  7162 root     Ds    3.2 fio               ext4_file_write
 7159  7164 root     Ds    3.2 fio               ext4_file_write
```

Processes doing I/O can be in waiting state

Kernel function process is sleeping on

vmstat plots

THOMAS KRENN®
server.hosting.customized.



Drawing interrupts and context switches

https://clusterbuffer.wordpress.com/admin-tools/vmstat-plotter/

But we are not satisfied with summaries and overviews…

What is PID 9059 doing?

pidstat (part of sysstat)

server.hosting.customized.

- Report statistics for tasks being managed by kernel
- CPU bound → identify peak activity

```
$ top -b -n 1 -d 2 -o %CPU | head
[...]
  PID USER       PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 9059 gschoenb   20   0   47532  21132   2444 R  96,9  0,3   0:02.14 python
    1 root       20   0   33880   3256   1500 S   0,0  0,0   0:02.35 init
$ pidstat -p 9059 -u 1 -l
Linux 3.13.0-48-generic (X220) 2015-04-15      _x86_64_   (4 CPU)

10:11:04      UID        PID    %usr %system  %guest    %CPU   CPU  Command
10:11:05     1000       9059  100,00    0,00    0,00  100,00     0  python ijk-matrix.py
-i matrix.in
10:11:06     1000       9059  100,00    0,00    0,00  100,00     0  python ijk-matrix.py
-i matrix.in
10:11:07     1000       9059  10                                 0  python ijk-matrix.py
-i matrix.in
```

Even check command line arguments (“-l”) !

pidstat

_ I/O bound → device report

```
# mpstat -P ALL 1
10:25:31      CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
10:25:32      all   14,88    0,00    9,40   13,84    0,00    1,04    0,00    0,00    0,00   60,84
10:25:32        0   22,45    0,00    1,02    0,00    0,00    0,00    0,00    0,00    0,00   76,53
10:25:32        1   13,73    0,00   34,31   51,96    0,00    0,00    0,00    0,00    0,00    0,00
10:25:32        2   17,86    0,00    0,00    0,00    0,00    3,[illegible]                                  78,57
10:25:32        3    6,12    0,00    0,00    0,00    0,00 [illegible]                                  93,88
# pidstat -d 1
Linux 3.13.0-48-generic (X220)  2015-04-15      _x86_64_    (4 CPU)

10:26:35      UID       PID   kB_rd/s   kB_wr/s kB_ccwr/s  Command
10:26:36        0      9208      0,00   2303,85      0,00  fio
10:26:36        0      9209      0,00   2996,15      0,00  fio
10:26:36        0      9210      0,00   2023,08      0,00  fio
10:26:36        0      9211      0,00   1284,62      0,00  fio
```

Which process is causing %iowait?

Device report reveals command and I/O

pidstat

_ How much memory is PID 8461 using?

- Major faults require I/O operations, good indicator you need more RAM!

```
# pidstat -r -p 8461 1 3
Linux 3.13.0-49-generic (X220)  2015-04-21    _x86_64_ (4 CPU)

10:09:06      UID       PID  minflt/s  majflt/s     VSZ    RSS   %MEM  Command
10:09:07     1000      8461      8,00      0,00 2018384 786688   9,76  firefox
10:09:08     1000      8461     11,00      0,00 2018384 786688   9,76  firefox
10:09:09     1000      8461     23,00      0,00 2018448 786892   9,77  firefox
Average:     1000      8461     14,00      0,00 2018405 78675    9,77  firefox
```

Minor and major page faults

Current used share of physical memory

iostat (part of sysstat)

- I/O subsystem statistics
- CPU or device utilization report
- Without argument → summary since boot
  - Skip that with -y option

```
# iostat
Linux 3.13.0-48-generic (X220)  2015-04-15      _x86_64_   (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          16,16    0,09    4,79    0,46    0,00   78,50

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda              83,80        41,64       531,43   22375057  285581196
```

iostat

_ CPU util report → `%iowait`

_ Not really reliable → `%iowait` is some kind of `%idle` time

```
# taskset 1 fio -rw=randwrite [...] &
# iostat -y -c 1 3
[…]
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          17,32    0,00    6,56   13,65    0,00   62,47
# taskset 1 sh -c "while true; do true; done" &
# iostat -y -c 1 3
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          35,59    0,00    7,02    0,00    0,00   57,39
```

http://www.percona.com/blog/2014/06/03/trust-vmstat-iowait-numbers/

`iostat`

_ Extended device util report ("-x") → `%util`

- _ `man iostat` → *... for devices serving request serially, for parallel processing (RAID arrays and SSDs), this number does not reflect their performance limits.*

_ In theory

- _ 94,4% `util` 23032 IOPS
- _ 99,6% `util` 24300 IOPS

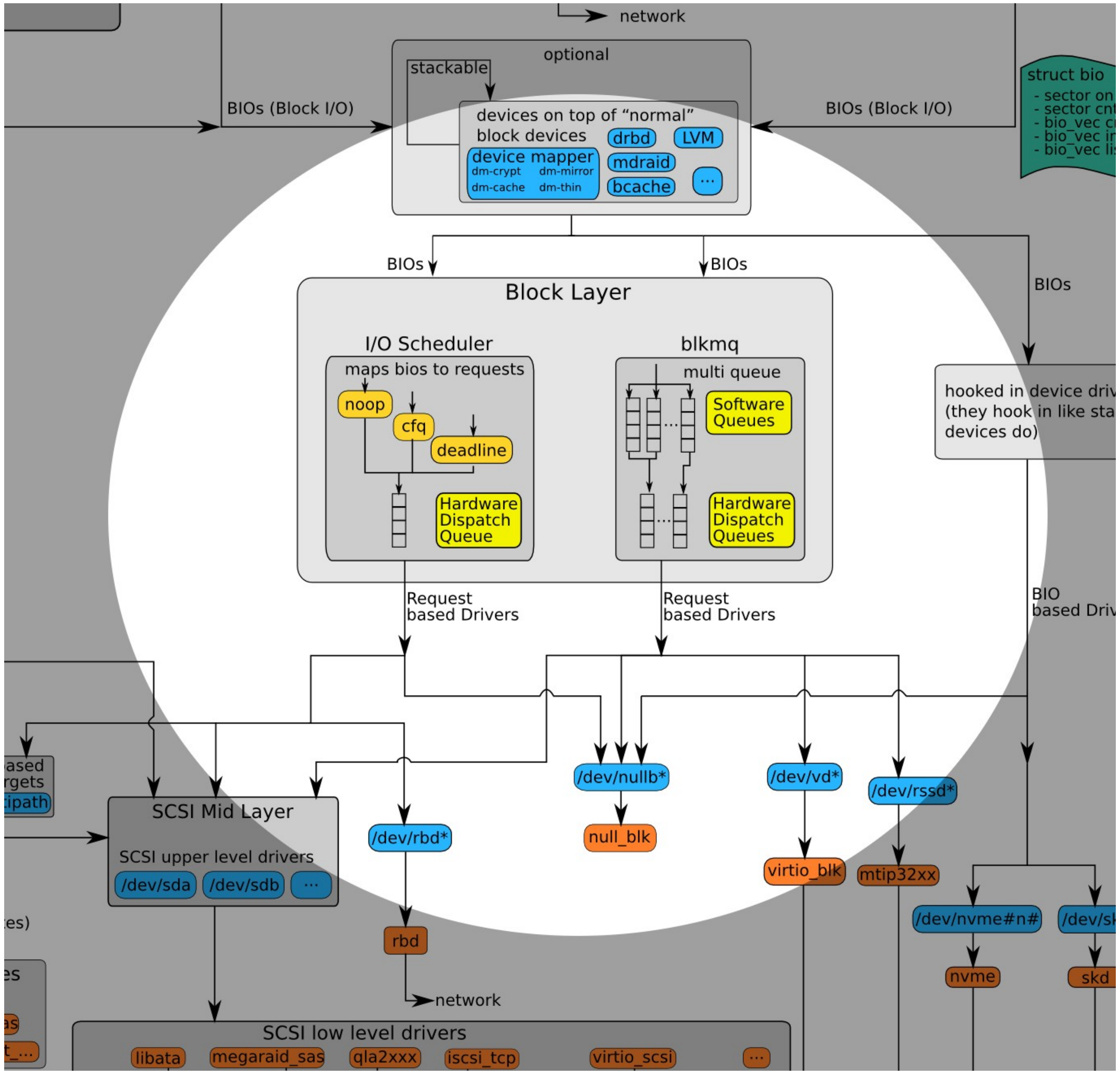
iostat

```
# iostat -y -d -x 1 3
Linux 3.13.0-48-generic (X220) 2015-04-15      _x86_64_   (4 CPU)

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     2,00    0,00 23032,00     0,00 92136,00     8,00     2,90    0,13
0,00    0,13   0,04  94,40
# iostat -y -d -x 1 3
Linux 3.13.0-48-generic (X220) 2015-04-15      _x86_64_   (4 CPU)

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  2917,00    0,00 43175,00     0,00 184[illegible]0,00     8,55   135,75    3,15
0,00    3,15   0,02  99,60
```

Only 5% `util` increase,
but IOPS nearly doubled!

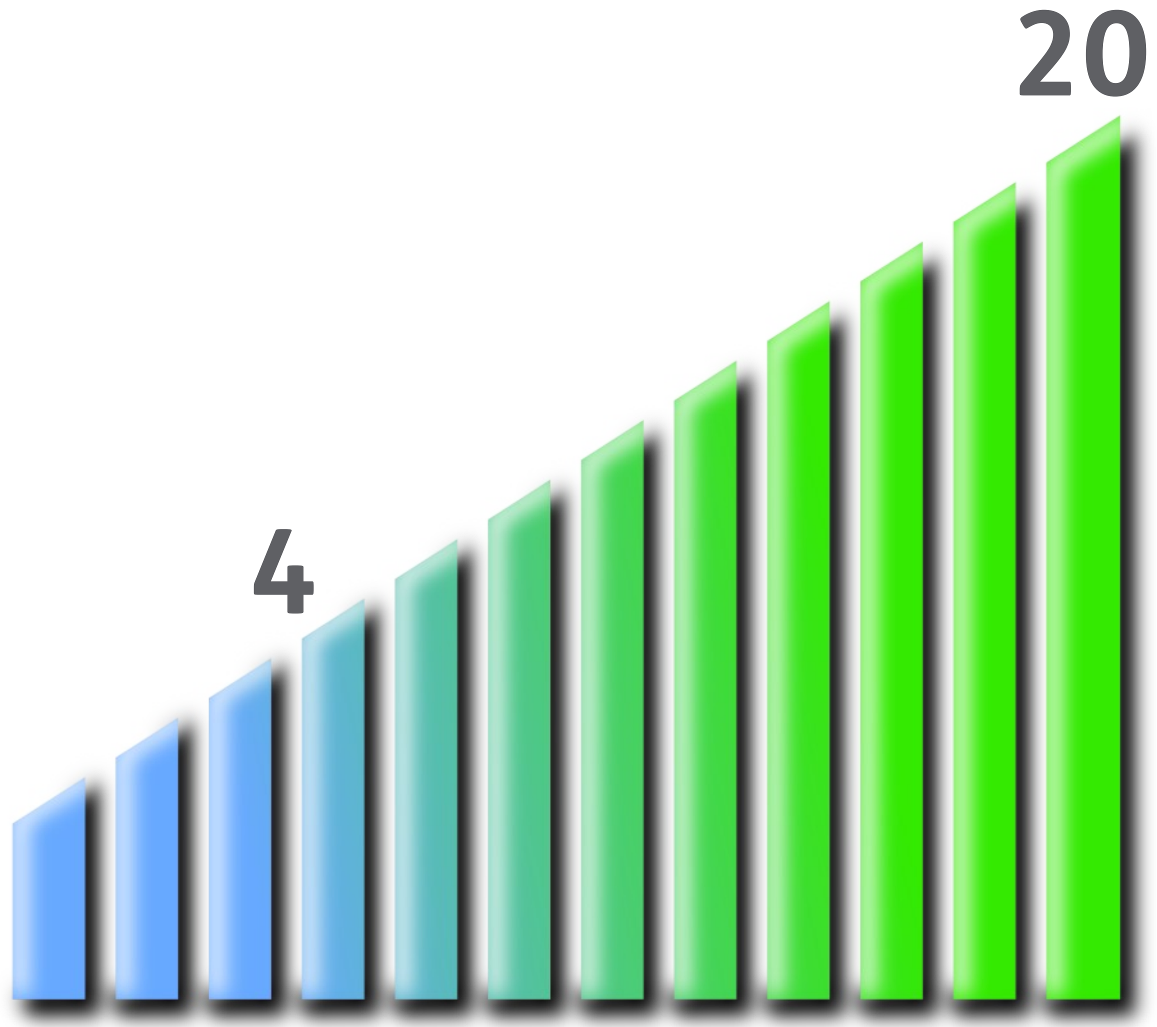


https://www.thomas-krenn.com/de/wiki/Linux_Storage_Stack_Diagramm

iostat

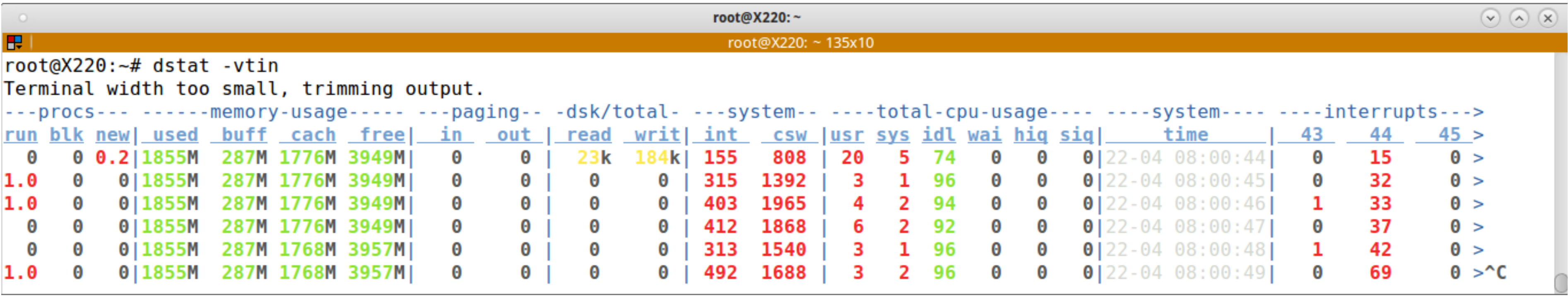
- `avgqu-sz` Avg. queue length of requests issued
  - `(delta[time_in_queue] / interval) / 1000.0`
    - `time_in_queue` Requets waiting for device, effected by `in_flight`
- `await` Avg. time requests being served
  - `delta[read_ticks + write_ticks] / delta[read_IOs + write_Ios]`
  - `ticks` also effected by `in_flight`
- Therefore serving more requests while `await` is not increasing, is a good performance indicator

- Monitoring IO Performance using iostat and pt-diskstats
- Block layer statistics


4
20

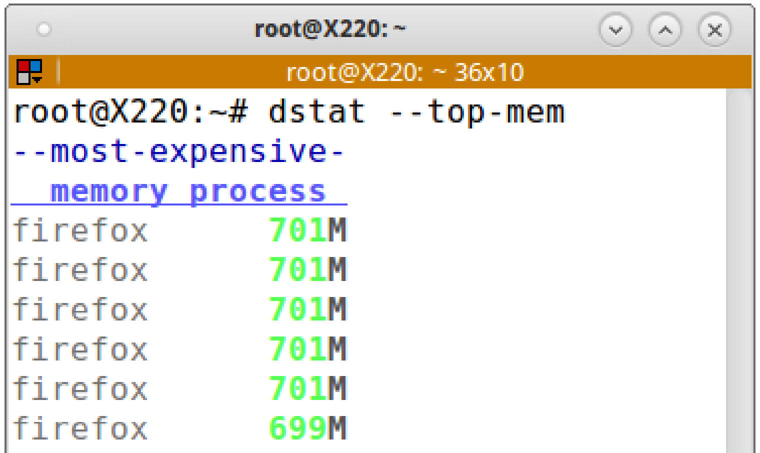
dstat

- Combines several classical tools
  - Prints metrics and uses colors
  - Has a plugin concept


root@X220: ~

root@X220: ~ 135x10

```
root@X220:~# dstat -vtin
Terminal width too small, trimming output.
---procs--- ------memory-usage----- ---paging-- -dsk/total- ---system-- ----total-cpu-usage---- ----system---- ----interrupts--->
run blk new| used  buff  cach  free|  in   out | read  writ| int   csw |usr sys idl wai hiq siq|     time     |  43    44    45 >
  0   0 0.2|1855M  287M 1776M 3949M|   0     0 |  23k  184k| 155   808 | 20   5  74   0   0   0|22-04 08:00:44|   0    15     0 >
1.0   0   0|1855M  287M 1776M 3949M|   0     0 |   0     0 | 315  1392 |  3   1  96   0   0   0|22-04 08:00:45|   0    32     0 >
1.0   0   0|1855M  287M 1776M 3949M|   0     0 |   0     0 | 403  1965 |  4   2  94   0   0   0|22-04 08:00:46|   1    33     0 >
  0   0   0|1855M  287M 1776M 3949M|   0     0 |   0     0 | 412  1868 |  6   2  92   0   0   0|22-04 08:00:47|   0    37     0 >
  0   0   0|1855M  287M 1768M 3957M|   0     0 |   0     0 | 313  1540 |  3   1  96   0   0   0|22-04 08:00:48|   1    42     0 >
1.0   0   0|1855M  287M 1768M 3957M|   0     0 |   0     0 | 492  1688 |  3   2  96   0   0   0|22-04 08:00:49|   0    69     0 >^C
```


root@X220: ~

root@X220: ~ 36x10

```
root@X220:~# dstat --top-mem
--most-expensive-
  memory process
firefox       701M
firefox       701M
firefox       701M
firefox       701M
firefox       701M
firefox       699M
```

nicstat

- Print network device statistics
  - `%Util` depends on speed and duplex mode
  - `Sat` also takes errors into account

Check if your network is saturated, `Drops` can be an indicator!

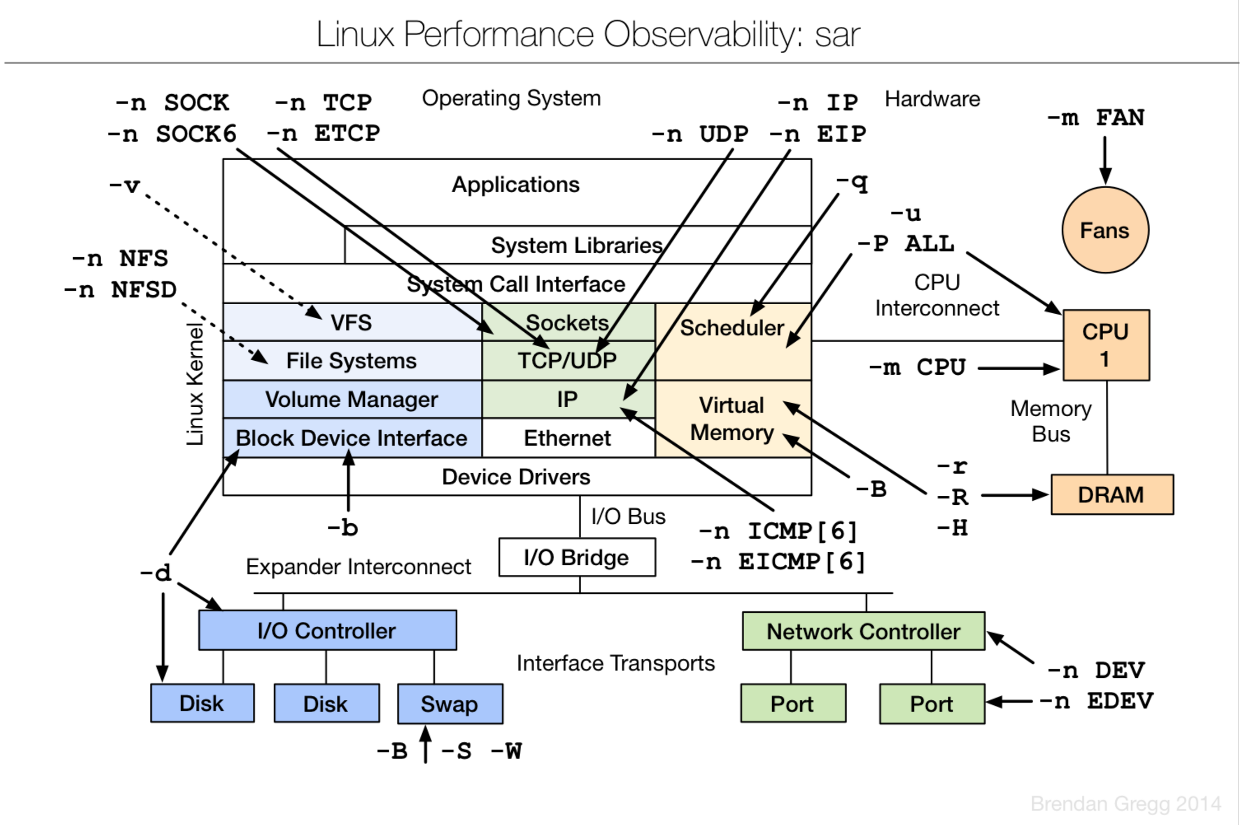
```
# nicstat -l
Int        Loopback   Mbit/s Duplex State
vboxnet0         No        0   unkn    up
eth0             No     1000   full    up
lo              Yes        -   unkn    up
wlan0            No        0   unkn    up
# nicstat -i eth0 1 5
    Time      Int   rKB/s   wKB/s   rPk/s   wPk/s    rAvs    wAvs %Util    Sat
14:52:21     eth0    3.08    0.36    3.13    2.48  1007.6   149.4  0.00   0.05
14:52:22     eth0   19.89    1.23   16.98   17.97  1199.6   70.00  0.02   0.00
14:52:23     eth0   21.42    1.09   21.99   16.00   997.1   70.00  0.02   0.00
# nicstat -i eth0 -t 1 2
14:57:36    InKB   OutKB   InSeg  OutSeg Reset  AttF %ReTX InConn OutCon Drops
TCP         0.00    0.00    2.88    2.51  0.02  0.00 0.000   0.00   0.04  0.00
14:57:37    InKB   OutKB   InSeg  OutSeg Reset  AttF %ReTX InConn OutCon Drops
TCP         0.00    0.00    0.00    0.00  0.00  0.00 0.000   0.00   0.00  0.00
```

Do you have a history of your system's performance data?

yes	no

sar (part of sysstat)

- It's easy with system activity reporter
  - sar, sadc, sa1 and sa2, sadf

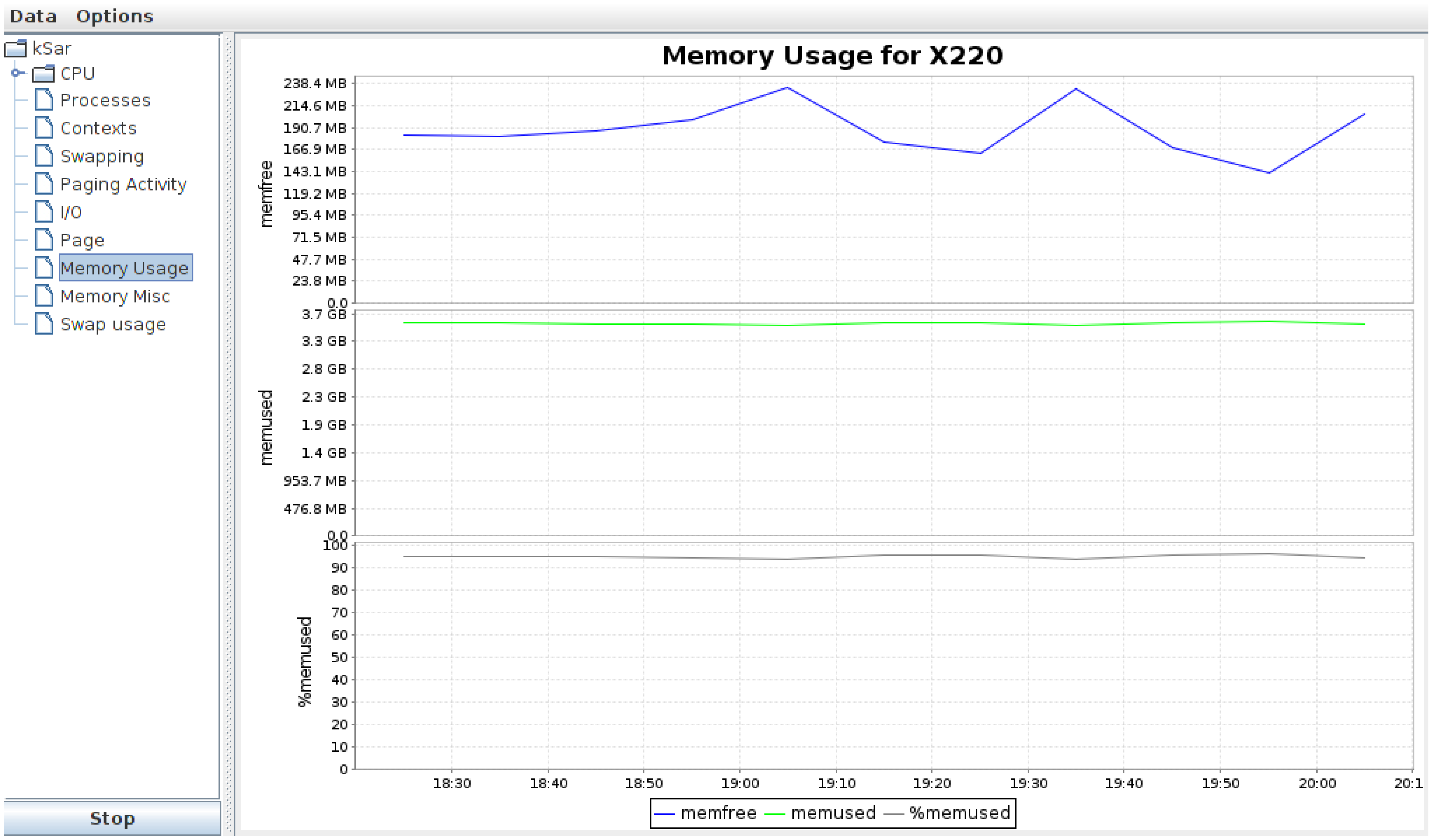


http://www.brendangregg.com/Perf/linux_observability_sar.png

ksar

Mitigates character encoding and number format problems

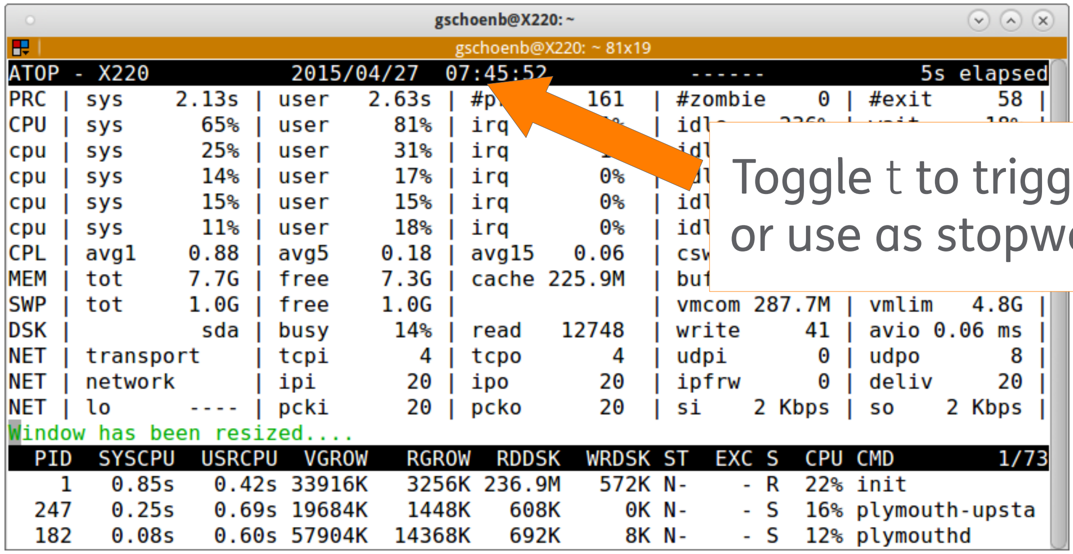
```
_ LC_ALL=POSIX sar -A -f sa10 > ksar.out.txt
```



atop

_ Sets up a cronjob per default

```
$ grep start -A 2 /etc/cron.d/atop
# start atop daily at midnight
0 0 * * * root invoke-rc.d atop _cron
$ ls /var/log/atop/atop_201504*
/var/log/atop/atop_20150401  /var/log/atop/atop_20150409  /var/log/atop/atop_20150422
/var/log/atop/atop_20150402  /var/log/atop/atop_20150420  /var/log/atop/atop_20150424
/var/log/atop/atop_20150408  /var/log/atop/atop_20150421  /var/log/atop/atop_20150427
$ atop -r /var/log/atop/atop_20150427
```

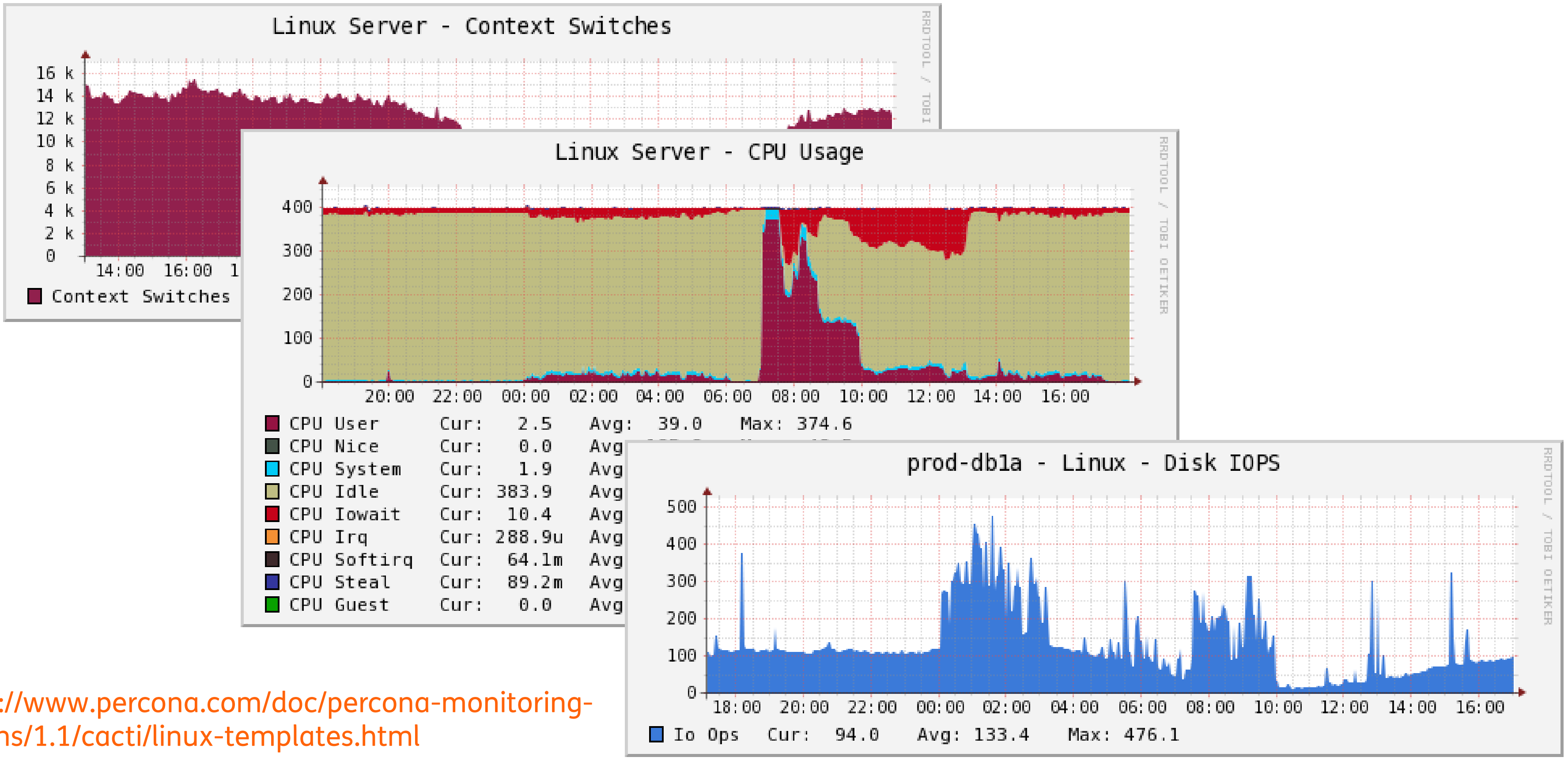


```
ATOP - X220            2015/04/27  07:45:52            ------            5s elapsed
PRC | sys    2.13s | user   2.63s | #p        161  | #zombie    0 | #exit     58 |
CPU | sys      65% | user     81% | irq            | idl
cpu | sys      25% | user     31% | irq            | idl
cpu | sys      14% | user     17% | irq        0%  | idl
cpu | sys      15% | user     15% | irq        0%  | idl
cpu | sys      11% | user     18% | irq        0%  | idl
CPL | avg1    0.88 | avg5    0.18 | avg15   0.06   | csw
MEM | tot     7.7G | free    7.3G | cache 225.9M   | buf
SWP | tot     1.0G | free    1.0G |                | vmcom 287.7M | vmlim   4.8G |
DSK |          sda | busy     14% | read   12748   | write     41 | avio 0.06 ms |
NET | transport    | tcpi       4 | tcpo       4   | udpi       0 | udpo       8 |
NET | network      | ipi       20 | ipo       20   | ipfrw      0 | deliv     20 |
NET | lo      ---- | pcki      20 | pcko      20   | si    2 Kbps | so    2 Kbps |
Window has been resized....
  PID  SYSCPU  USRCPU  VGROW   RGROW  RDDSK  WRDSK ST  EXC S  CPU CMD          1/73
    1   0.85s   0.42s 33916K  3256K 236.9M  572K N-    - R  22% init
  247   0.25s   0.69s 19684K  1448K   608K    0K N-    - S  16% plymouth-upsta
  182   0.08s   0.60s 57904K 14368K   692K    8K N-    - S  12% plymouthd
```

Toggle t to trigger a new sample or use as stopwatch if interval=0

Percona Cacti Template

- Percona Linux Monitoring Template for Cacti
- generate many graphs easily



```
🟥 Context Switches
```

```
■ CPU User      Cur:   2.5    Avg:  39.0    Max: 374.6
■ CPU Nice      Cur:   0.0    Avg
■ CPU System    Cur:   1.9    Avg
■ CPU Idle      Cur: 383.9    Avg
■ CPU Iowait    Cur:  10.4    Avg
■ CPU Irq       Cur: 288.9u   Avg
■ CPU Softirq   Cur:  64.1m   Avg
■ CPU Steal     Cur:  89.2m   Avg
■ CPU Guest     Cur:   0.0    Avg
```

```
🟦 Io Ops   Cur:  94.0   Avg: 133.4   Max: 476.1
```

https://www.percona.com/doc/percona-monitoring-plugins/1.1/cacti/linux-templates.html

Agenda

- Collect Statistics
  - Sysstat Package
    - iostat
    - pidstat
  - sar and sadc
  - Percona Cacti Template
- Watch online
  - top
  - iotop
  - iftop
- Tracing
  - perf_events
  - ftrace
  - perf-tools
  - Flame graphs

top

- System summary at beginning
- Per process metrics afterwards
  - Default sorted by CPU usage

1, 5 and 15 min load average

```
$ top -b -n 1| head -15
top - 15:33:50 up 3 days, 19:02,  3 users,  load average: 0.13, 0.51, 0.59
Tasks: 668 total,   1 running, 667 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.5%us,  0.3%sy,  0.1%ni, 98.1%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  132009356k total, 23457172k used, 108552184k free,  1600120k buffers
Swap:  3904444k total,        0k used,  3904444k free, 12682188k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
29276 root      20   0  6928 3488  668 S   19  0.0  22:55.72 ossec-syscheckd
 1193 gschoenb  20   0 17728 1740  936 R    4  0.0   0:00.02 top
11257 root      20   0 22640 2636 1840 S    4  0.0  70:38.88 openvpn
19907 www-data  20   0  197m  61m  52m S    4  0.0   0:06.18 apache2
  775 root      20   0     0    0    0 S    2  0.0   8:03.13 md3_raid10
 3712 root      39  19     0    0    0 S    2  0.0  22:45.85 kipmi0
12807 root      -3   0     0    0    0 S    2  0.0   6:20.30 drbd2_asender
 18653 root      20   0     0    0    0 S    2  0.0  12:40.19 drbd1_receiver
```

top

THOMAS KRENN®
server.hosting.customized.

- Memory usage
  - `VIRT` The total size of virtual memory for the process
    - Also including e.g. shared libraries, not already mapped `heap` or `swap`
    - how much memory the program is able to access at the moment
  - `RES` How many blocks are really allocated and mapped to address space → resident
    - how much actual physical memory a process is consuming
  - `SHR`
    - how much of the VIRT size is actually sharable

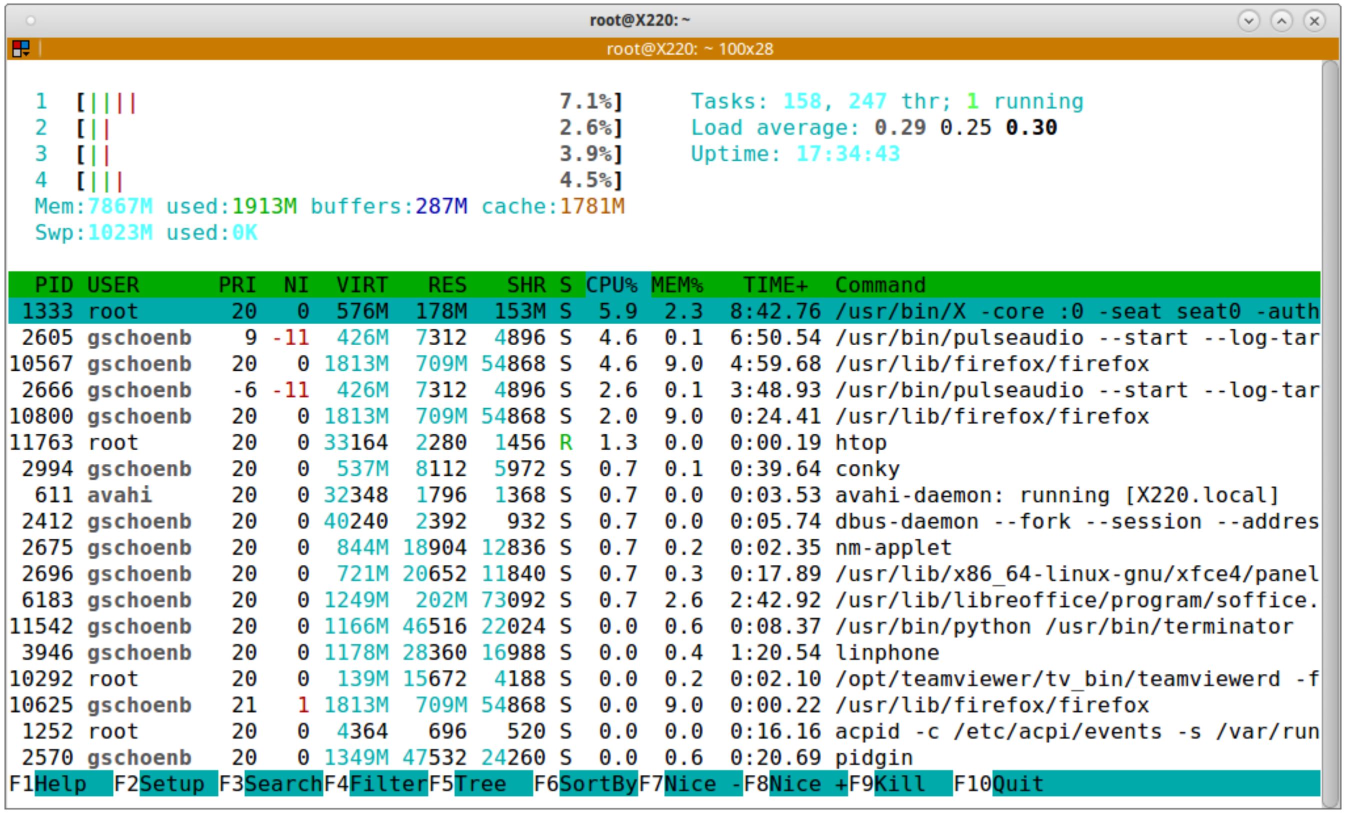
- https://www.linux.com/learn/tutorials/42048-uncover-the-meaning-of-tops-statistics
- http://www.linuxdevcenter.com/pub/a/linux/2006/11/30/linux-out-of-memory.html

`top`

- Can consume resources on it's own
- Toggle `f` and select fields, e.g. `SWAP`
- `-u` let's you see processes from a user
- Toggle `k` to kill a PID
- Toggle `r` to renice a PID
- But
  - `top` can miss short living processes
  - high `%CPU` → so what?
    - Keep an eye on the tracing part

htop

- „Super advanced“ `top`
  - Uses colors, views can be customized



root@X220: ~

root@X220: ~ 100x28

```
  1  [||||                                       7.1%]     Tasks: 158, 247 thr; 1 running
  2  [||                                         2.6%]     Load average: 0.29 0.25 0.30
  3  [||                                         3.9%]     Uptime: 17:34:43
  4  [|||                                        4.5%]
  Mem:7867M used:1913M buffers:287M cache:1781M
  Swp:1023M used:0K

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 1333 root       20   0  576M  178M  153M S  5.9  2.3  8:42.76 /usr/bin/X -core :0 -seat seat0 -auth
 2605 gschoenb    9 -11  426M  7312  4896 S  4.6  0.1  6:50.54 /usr/bin/pulseaudio --start --log-tar
10567 gschoenb   20   0 1813M  709M 54868 S  4.6  9.0  4:59.68 /usr/lib/firefox/firefox
 2666 gschoenb   -6 -11  426M  7312  4896 S  2.6  0.1  3:48.93 /usr/bin/pulseaudio --start --log-tar
10800 gschoenb   20   0 1813M  709M 54868 S  2.0  9.0  0:24.41 /usr/lib/firefox/firefox
11763 root       20   0 33164  2280  1456 R  1.3  0.0  0:00.19 htop
 2994 gschoenb   20   0  537M  8112  5972 S  0.7  0.1  0:39.64 conky
  611 avahi      20   0 32348  1796  1368 S  0.7  0.0  0:03.53 avahi-daemon: running [X220.local]
 2412 gschoenb   20   0 40240  2392   932 S  0.7  0.0  0:05.74 dbus-daemon --fork --session --addres
 2675 gschoenb   20   0  844M 18904 12836 S  0.7  0.2  0:02.35 nm-applet
 2696 gschoenb   20   0  721M 20652 11840 S  0.7  0.3  0:17.89 /usr/lib/x86_64-linux-gnu/xfce4/panel
 6183 gschoenb   20   0 1249M  202M 73092 S  0.7  2.6  2:42.92 /usr/lib/libreoffice/program/soffice.
11542 gschoenb   20   0 1166M 46516 22024 S  0.0  0.6  0:08.37 /usr/bin/python /usr/bin/terminator
 3946 gschoenb   20   0 1178M 28360 16988 S  0.0  0.4  1:20.54 linphone
10292 root       20   0  139M 15672  4188 S  0.0  0.2  0:02.10 /opt/teamviewer/tv_bin/teamviewerd -f
10625 gschoenb   21   1 1813M  709M 54868 S  0.0  9.0  0:00.22 /usr/lib/firefox/firefox
 1252 root       20   0  4364   696   520 S  0.0  0.0  0:16.16 acpid -c /etc/acpi/events -s /var/run
 2570 gschoenb   20   0 1349M 47532 24260 S  0.0  0.6  0:20.69 pidgin
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

iotop

- Simple top like I/O monitor
- Which process is causing I/O
  - Filtering specific PID is possible

Show writes, reads and command in realtime

```
# iotop -o -b
Total DISK READ :       0.00 B/s | Total DISK WRITE :      63.94 M/s
Actual DISK READ:       0.00 B/s | Actual DISK WRITE:      63.90 M/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN      IO    COMMAND
19153 be/4 root        0.00 B/s   63.89 M/s  0.00 % 75.44 % fio --rw=randwrite --name=test
--filename=test.fio --size=300M --direct=1 --bs=4k
17715 be/4 gschoenb    0.00 B/s   46.18 K/s  0.00 %  0.00 % firefox [mozStorage #1]
# iotop -o -b
Total DISK READ :      69.02 M/s | Total DISK WRITE :      65.92 K/s
Actual DISK READ:      69.02 M/s | Actual DISK WRITE:     345.12 K/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN      IO    COMMAND
19176 be/4 root       69.02 M/s    0.00 B/s  0.00 % 88.28 % fio --rw=read --name=test
--filename=test.fio --size=300M --direct=1 --bs=8k
```

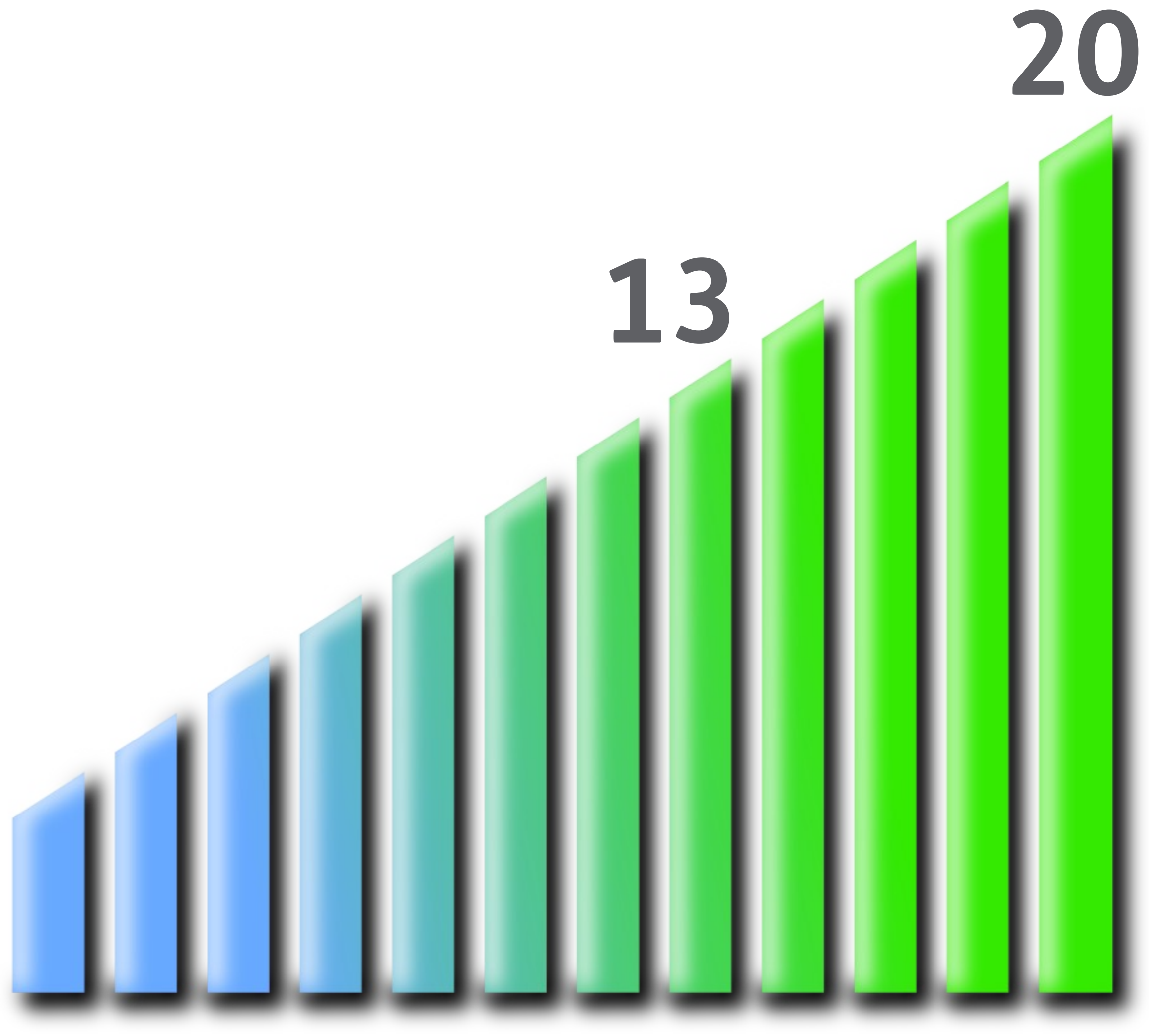
Bandwidth live usage

- iftop
  - Per interface usage
- nethogs
  - Per process

```
NetHogs version 0.8.0

  PID USER     PROGRAM                                        DEV         SENT        RECEIVED
17692 gschoenb /usr/lib/firefox/firefox                       eth0       0.162         0.194 KB/sec
16585 root     /usr/bin/ssh                                   eth0       0.000         0.000 KB/sec
16611 gschoenb evolution                                      eth0       0.000         0.000 KB/sec
?     root     unknown TCP                                               0.000         0.000 KB/sec

  TOTAL                                                                  0.162         0.194 KB/sec
```


13
20

Agenda

- Collect Statistics
  - Sysstat Package
    - `iostat`
    - `pidstat`
  - `sar` and `sadc`
  - Percona Cacti Template
- Watch online
  - `atop`
  - `top`
  - `iotop`
  - `iftop`
- Tracing
  - `perf_events`
  - `ftrace`
  - `perf-tools`
  - Flame graphs

```
# whereis tracing
```

Profiling

- Create profile about usage characteristics
  - Count specific samples/events
  - Count objects
- Collecting statistics about tracepoints
  - Lines of kernel code with defined event
- Next slides focus on system profiling
  - `ftrace`
  - `perf_events` and `perf`

ftrace

- Part of the Linux kernel since 2.6.27 (2008)
- What is going on inside the kernel
- Common task is to trace events
- With `ftrace` configured, only `debugfs` is required

```
# cat /proc/sys/kernel/ftrace_enabled
1
# mount | grep debug
none on /sys/kernel/debug type debugfs (rw)
/sys/kernel/debug/tracing# cat available_tracers
blk mmiotrace function_graph wakeup_rt wakeup function nop
```

Most widely applicable, traces kernel function calls

ftrace

_ Interact with files in `sys`

```
#!/bin/bash

DEBUGFS=`grep debugfs /proc/mounts | awk '{ print $2; }'`

echo $$ > $DEBUGFS/tracing/set_ftrace_pid
echo function > $DEBUGFS/tracing/current_tracer
echo 1 > $DEBUGFS/tracing/tracing_on
$*
echo 0 > $DEBUGFS/tracing/tracing_on
```

View the recorded trace

```
# less /sys/kernel/debug/tracing/trace
```

_ Easier with `trace-cmd` → interface for `sys` files

perf_events and perf

_ Used to be called performance counters for Linux

_ A lot of updates for kernel 4.1

  _ https://lkml.org/lkml/2015/4/14/264

_ CPU performance counters, tracepoints, kprobes and uprobes

_ Per package with `linux-tools-common`

```
# which perf
/usr/bin/perf
# dpkg -S /usr/bin/perf
linux-tools-common: /usr/bin/perf
```

perf list

- perf list
  - Shows supported events

This also includes static tracepoints

```
# perf list | wc -l
1779
# perf list | grep Hardware
  cpu-cycles OR cycles                               [Hardware event]
  instructions                                       [Hardware event]
  cache-references                                   [Hardware event]
  cache-misses                                       [Hardware event]
  branch-instructions OR branches                    [Hardware event]
  branch-misses                                      [Hardware event]
  bus-cycles                                         [Hardware event]
  stalled-cycles-frontend OR idle-cycles-frontend    [Hardware event]
  stalled-cycles-backend OR idle-cycles-backend      [Hardware event]
  ref-cycles                                         [Hardware event]
  L1-dcache-loads                                    [Hardware cache event]
  L1-dcache-load-misses                              [Hardware cache event]
  L1-dcache-stores                                   [Hardware cache event]
  L1-dcache-store-misses                             [Hardware cache event]
```

Raw CPU counters

- Each CPU has it's own raw counters
  - They should be documented by the hardware manufacturer
    - https://download.01.org/perfmon/
- `libpfm4` is a nice way to find raw masks

```
# perf list | grep rNNN
  rNNN                                          [Raw hardware event descriptor]
# git clone git://perfmon2.git.sourceforge.net/gitroot/perfmon2/libpfm4
# cd libpfm4
# make
# cd examples/
# ./showevtinfo | grep LLC | grep MISSES
Name     : LLC_MISSES
[...]
# ./check_events LLC_MISSES | grep Codes
Codes           : 0x53412e
# perf stat -e r53412e sleep 5
```

Now we collect last level cache misses with the raw mask

Tracepoints

_ `perf` also has trace functionalities

```
# perf list | grep -i trace | wc -l
1716
```

- Filesystem
- Block layer
- Syscalls

```
# perf stat -e 'syscalls:sys_enter_mmap' ./helloWorld.out
Hello world!

 Performance counter stats for './helloWorld.out':

                 8 syscalls:sys_enter_mmap

       0,000556961 seconds time elapsed
```

perf stat

_ Get a counter summary

A way to compare performance of different algorithms

```
# perf stat python numpy-matrix.py -i matrix.in

 Performance counter stats for 'python numpy-matrix.py -i matrix.in':

       576,104221 task-clock (msec)              0,930 CPUs utilized
              319 context-switches          #    0,554 K/sec
                4 cpu-migrations            #    0,007 K/sec
            9.738 page-faults               #    0,017 M/sec
    1.743.664.199 cycles                    #    3,027 GHz                     [82,63%]
      831.364.029 stalled-cycles-frontend   #   47,68% frontend cycles idle    [83,75%]
      458.760.523 stalled-cycles-backend    #   26,31% backend  cycles idle    [67,26%]
    2.793.953.303 instructions              #    1,60  insns per cycle
                                            #    0,30  stalled cycles per insn [84,28%]
      573.342.473 branches                  #  995,206 M/sec                   [83,78%]
        3.586.249 branch-misses             #    0,63% of all branches         [82,70%]

      0,619482128 seconds time elapsed
```

perf record

- Record samples to a file
  - Can be post-processed with `perf report`
  - `-a` records on all CPUs
  - `-g` records call graphs
    - Install debug symbols

```
# perf record -a -g sleep 5
[ perf record: Woken up 4 times to write data ]
[ perf record: Captured and wrote 2.157 MB perf.data (~94254 samples) ]
```

Nice way to record what's currently running on all CPUs

perf report

_ Displays profile of a record
  _ Can be sorted and or filtered
  _ Shows all samples

```
# perf report -i perf.data.dd --stdio --showcpuutilization --sort comm,dso
[...]
# Overhead       sys       usr  Command      Shared Object
# ........  ........  ........  .......  .................
    95.00%    95.00%     0.00%       dd  [kernel.kallsyms]
                 |
                 |--33.22%-- _aesni_enc1
                 |          __ablk_encrypt
                 |          ablk_encrypt
                 |          crypt_scatterlist
                 |          crypt_extent
                 |          ecryptfs_encrypt_page
                 |          ecryptfs_write_end
                 |          generic_file_buffered_write
                 |          __generic_file_aio_write
                 |          generic_file_aio_write
                 |          do_sync_write
                 |          vfs_write
                 |          sys_write
                 |          system_call_fastpath
                 |          __GI___libc_write
                 |          0x415f65643d524550
                 |--9.11%-- _cond_resched
                 |          |
                 |          |--57.94%-- ext4_dirty_inode
                 |          |          __mark_inode_dirty
                 |          |          generic_write_end
                 |          |          ext4_da_write_end
                 |          |          generic_file_buffered_write
```

Command and shared object

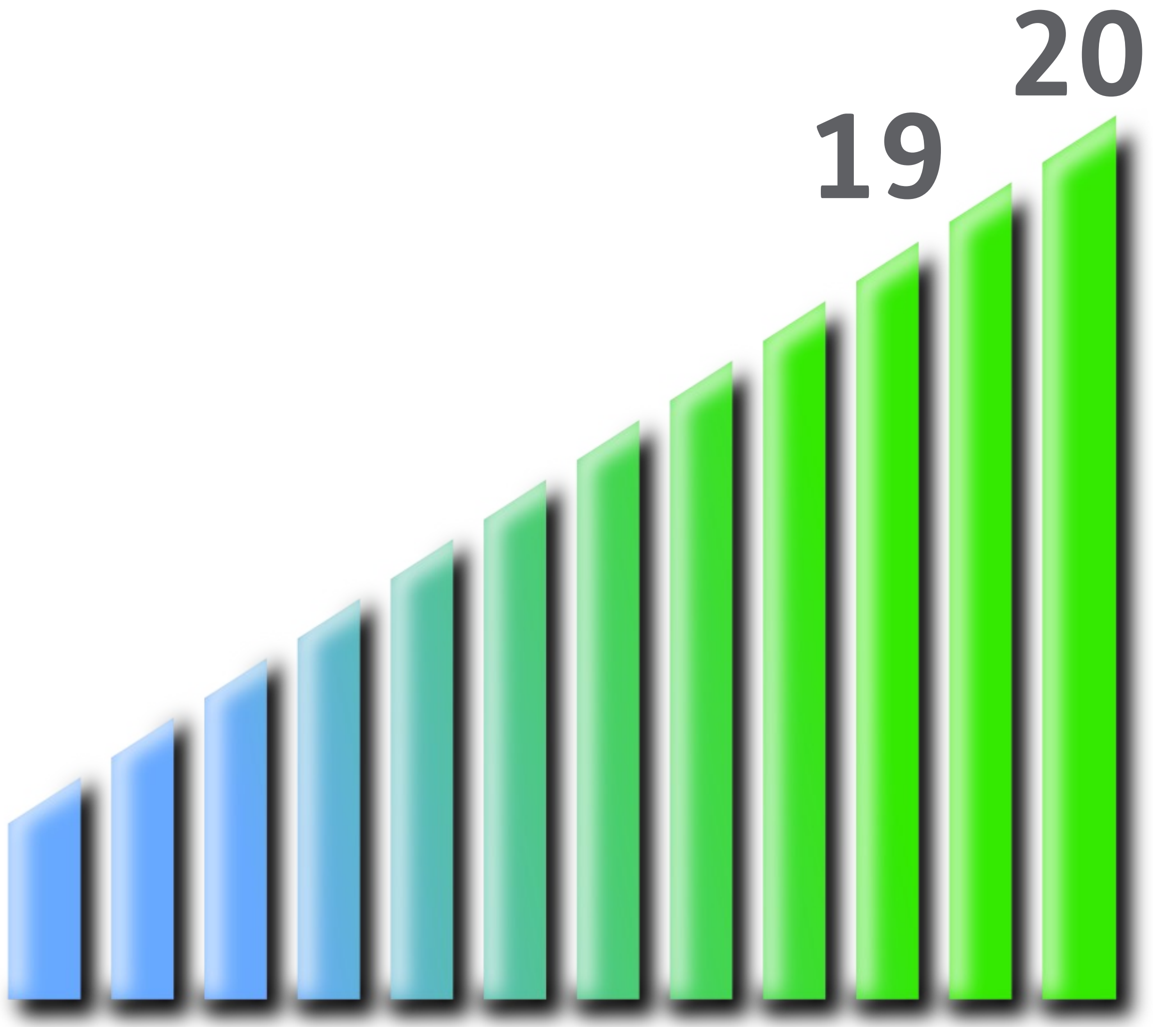
Traced method

dd writes data

perf-tools

- By Brendan Gregg
  - https://github.com/brendangregg/perf-tools
  - Mostly quick hacks, read Warnings!
- Using `perf_events` and `ftrace`
- Good examples what can be done with `perf` and `ftrace`
  - `iosnoop` Shows I/O access for commands, including latency
  - `cachestat` Linux page cache hit/miss statistics
  - `functrace` Count kernel functions matching wildcards

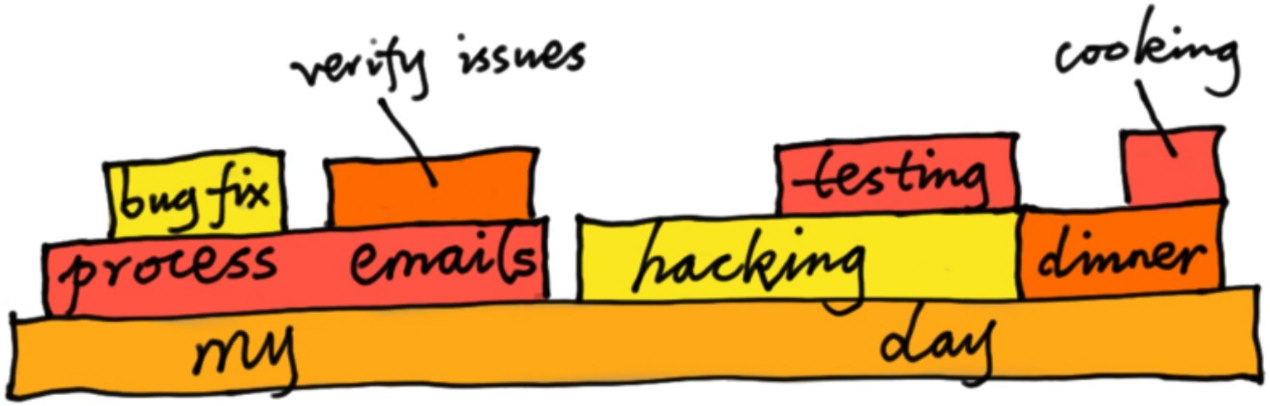
Nice, this are simple bash scripts!


19
20

```
# view flamegraph
```

Flamegraph

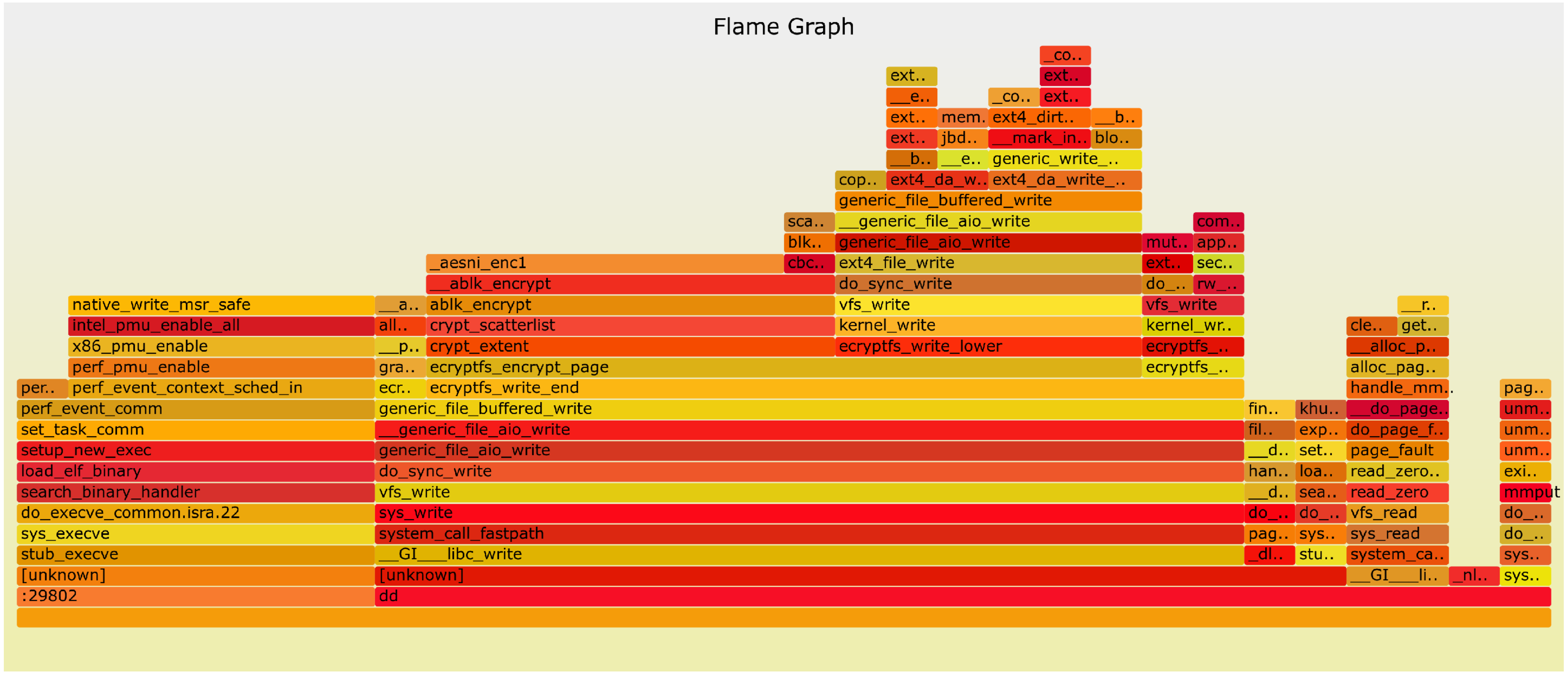
_ Visualization how resources are distributed among code



Powered by @agentzh, http://agentzh.org/misc/slides/yapc-na-2013-flame-graphs.pdf

Flamegraph

```
# perf record -g dd if=/dev/zero of=test.data count=1 bs=1M
# mv perf.data perf.data.dd
# perf script -i perf.data.dd | ./FlameGraph/stackcollapse-perf.pl > out.dd.folded
# ./FlameGraph/flamegraph.pl out.dd.folded > out.perf.dd.svg
```



Thanks for your attention!

_ cmitasch AT thomas-krenn.com
_ @cmitasch

server.hosting.customized.