GNU/Linux Performance tools: Profiling Rust

Martina Balintova

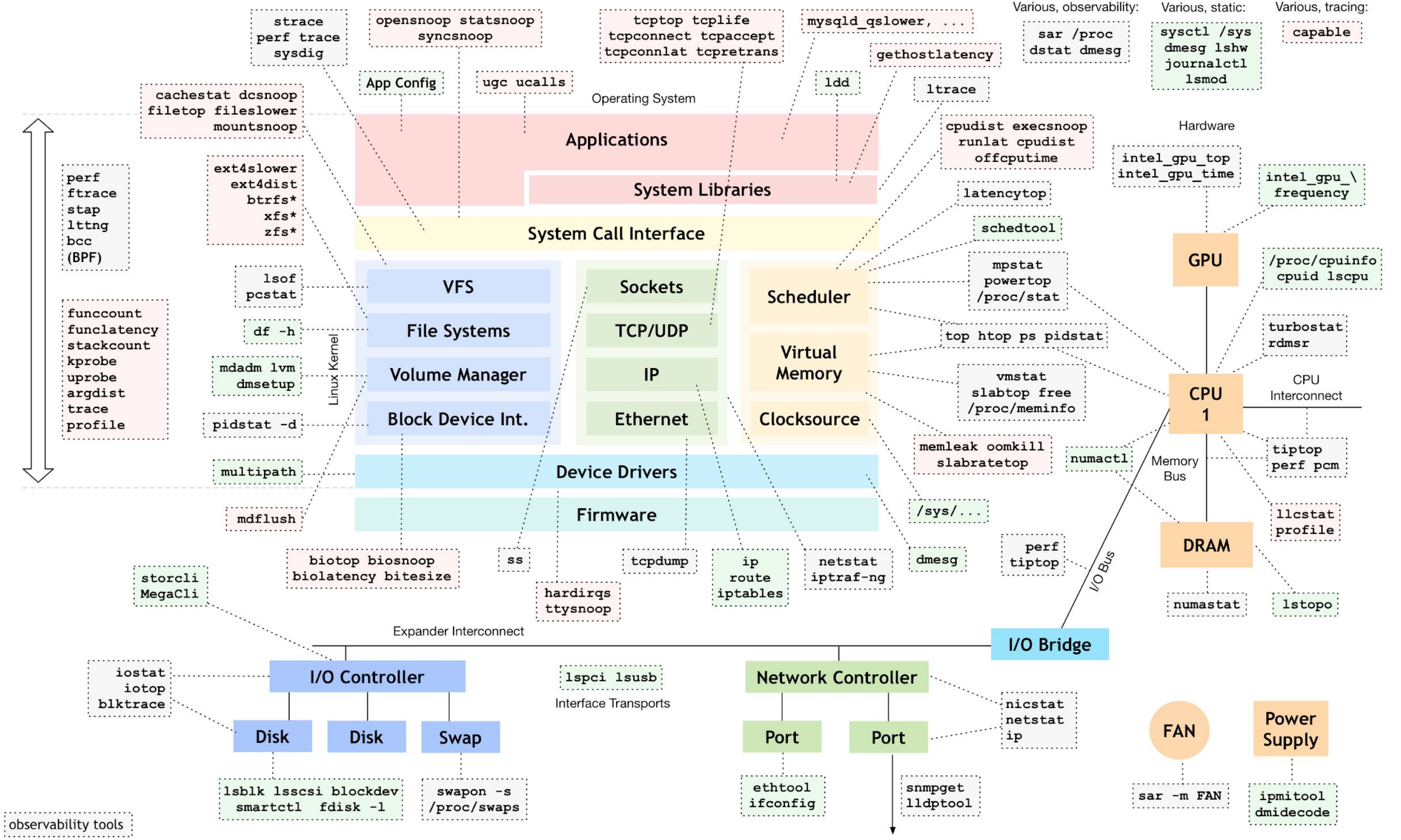
Couple of definitions

Benchmarking
<->
Performance measurements
<->
Tuning
<->
Configuration checks

- Memory performance (memory allocation, memory leaks, memory stalls - eg fetches for cache)
- Lock contention (spin loops, blocking wait)
- CPU utilization
- Multithreading
- Disk IO
- Network IO
- ...

Sampling <-> Counting

Linux Performance Tools



Operating System

Hardware

Linux Kernel

Applications

System Libraries

System Call Interface

VFS

File Systems

Volume Manager

Block Device Int.

Sockets

TCP/UDP

IP

Ethernet

Scheduler

Virtual Memory

Clocksource

Device Drivers

Firmware

strace perf trace sysdig

opensnoop statsnoop syncsnoop

tcptop tcplife tcpconnect tcpaccept tcpconnlat tcpretrans

mysqld_qslower, ...

gethostlatency

App Config

ugc ucalls

ldd

ltrace

cachestat dcsnoop filetop fileslower mountsnoop

cpudist execsnoop runlat cpudist offcputime

perf ftrace stap lttng bcc (BPF)

ext4slower ext4dist btrfs* xfs* zfs*

latencytop

schedtool

lsof pcstat

mpstat powertop /proc/stat

df -h

top htop ps pidstat

funccount funclatency stackcount kprobe uprobe argdist trace profile

mdadm lvm dmsetup

vmstat slabtop free /proc/meminfo

pidstat -d

memleak oomkill slabratetop

multipath

/sys/...

mdflush

biotop biosnoop biolatency bitesize

ss

hardirqs ttysnoop

tcpdump

ip route iptables

netstat iptraf-ng

dmesg

Various, observability:

sar /proc dstat dmesg

Various, static:

sysctl /sys dmesg lshw journalctl lsmod

Various, tracing:

capable

intel_gpu_top intel_gpu_time

intel_gpu_\ frequency

GPU

/proc/cpuinfo cpuid lscpu

turbostat rdmsr

CPU 1

CPU Interconnect

numactl

Memory Bus

tiptop perf pcm

llcstat profile

DRAM

perf tiptop

I/O Bus

numastat

lstopo

storcli MegaCli

Expander Interconnect

I/O Bridge

iostat iotop blktrace

I/O Controller

lspci lsusb

Interface Transports

Network Controller

nicstat netstat ip

Disk

Disk

Swap

Port

Port

FAN

Power Supply

lsblk lsscsi blockdev smartctl fdisk -l

swapon -s /proc/swaps

ethtool ifconfig

snmpget lldptool

sar -m FAN

ipmitool dmidecode

observability tools

static performance tools

these can observe the state of the system at rest, without load

perf-tools/bcc tracing tools

https://github.com/brendangregg/perf-tools https://github.com/iovisor/bcc

style inspired by reddit.com/u/redct

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

Poor man's profiling

- Run the binary (compile with debug symbols) to analyze under debugger
- Periodically stop the execution and get a backtrace
- Tools:
    - `gdb`
    - `lldb`
    - `(VS Code)`
    - ....

```
ad average: 0.03, 0.47, 0.33
ing,    0 stopped,    0 zombie

 SHR S  %CPU %MEM     TIME+ COMMAND
1376 S        0.0  0:00.58 init [4]
2300 S        0.0  0:01.01   `- /sbin/udevd --daemon
2344 S        0.0            `- /usr/sbin/cgmanager --daemon
1356 S        0.0            `- /usr/sbin/syslogd
     S        0.0            `- /usr/sbin/klogd -c 3 -x
   4 S        0.0            `- /sbin/dhcpcd -t 10 eth0
2184 S        0.0  0:00.13   `- /usr/bin/dbus-daemon --system
1436 S        0.0            `- /usr/sbin/ine
     S        0.0            `- /usr/sbin/ssh
     S        0.0  0:00.01   `- /usr/sbin/acp
4720 S        0.0  0:00.02   `- /usr/sbin/console-kit-da
I S    VIRT    RES   UID COMMAND
0 S    4368   1468     0 init [4]
0 S   36100   3804     0  `- /sbin/udevd --daemon
0 S   23768   2536     0  `- /usr/sbin/cgmanager --daemon
0 S    6504   1480     0  `- /usr/sbin/syslogd
0 S    4376     84     0  `- /usr/sbin/klogd -c 3 -x
0 S    6688    224     0  `- /sbin/dhcpcd -t 10 eth0
0 S   20332   2800    81  `- /usr/bin/dbus-dae   n --system
0 S    6488   1540     0  `- /usr/sbin/inetd
0 S   28676    428     0  `- /usr/sbin/sshd
0 S    4392    120     0  `- /usr/sbin/acpid
0 S  410392   7340     0  `- /usr/sbin/console-kit-daemon
A   SHR nMaj nDRT  %CPU COMMAND
6  1376   18            init [4]
2  2300                  `- /sbin/udevd --daemon
0  2344                  `- /usr/sbin/cgmanager --daemon
4  1356                  `- /usr/sbin/syslogd
4                        `- /usr/sbin/klogd -c 3 -x
2     4                  `- /sbin/dhcpcd -t 10 eth0
6  2184                  `- /usr/bin/dbus-dae   n --system
4  1436                  `- /usr/sbin/inetd
4                        `- /usr/sbin/sshd
4                        `- /usr/sbin/acpid
0  4720    7             `- /usr/sbin/console
    TIME+  %CPU %MEM S COMMAND
  0:00.58         0.0 S init [4]
  0:01.01         0.0 S  `- /sbin/udevd --daemon
                  0.0 S  `- /usr/sbin/cgmanager --daemon
                  0.0 S  `- /usr/sbin/syslogd
                  0.0 S  `- /usr/sbin/klogd -c 3 -x
                  0.0 S  `- /sbin/dhcpcd -t 10 eth0
  0:00.13         0.0 S  `- /usr/bin/dbus-daemon --system
                  0.0 S  `- /usr/sbin/inetd
                  0.0 S  `- /usr/sbin/sshd
  0:00.01         0.0 S  `- /usr/sbin/acpid
  0:00.02         0.0 S  `- /usr/sbin/console-kit-daemon
  0:00.05         0.1 S  `- /usr/lib/polkit-1/polkitd --no-debug
```

Real-time (interactive) list of linux processes.

- CPU summed across all CPUs, might lead to over 100%.
-

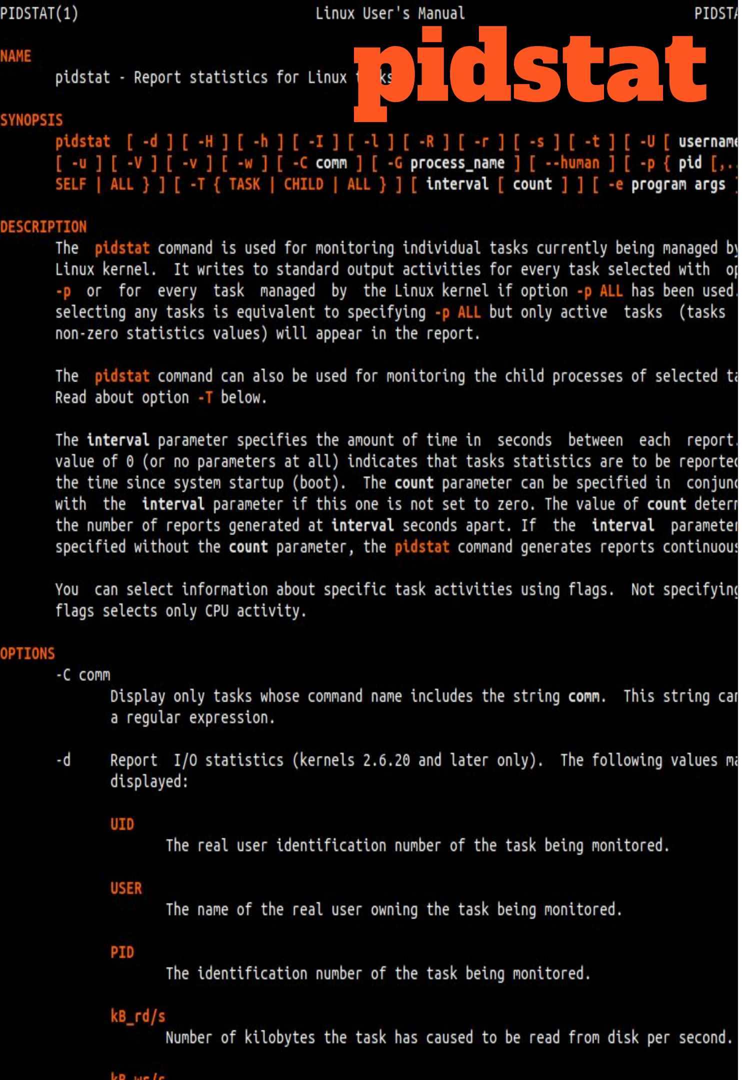
Usage:

- **top**

  ```
  zxcVm1t0
  A(w/a)
  ```

- **htop**
  - ‘s’ - trace processes’ system calls (strace)
  - ‘l’ - display open files for processes (lsof)
  - ‘L’ - trace library calls (ltrace) [only Debian]
- **iotop**
  - per process/thread I/O monitor

pidstat



```
PIDSTAT(1)                  Linux User's Manual                  PIDSTA

NAME
       pidstat - Report statistics for Linux tasks

SYNOPSIS
       pidstat  [ -d ] [ -H ] [ -h ] [ -I ] [ -l ] [ -R ] [ -r ] [ -s ] [ -t ] [ -U [ usernam
       [ -u ] [ -V ] [ -v ] [ -w ] [ -C comm ] [ -G process_name ] [ --human ] [ -p { pid [,.
       SELF | ALL } ] [ -T { TASK | CHILD | ALL } ] [ interval [ count ] ] [ -e program args

DESCRIPTION
       The  pidstat command is used for monitoring individual tasks currently being managed b
       Linux kernel.  It writes to standard output activities for every task selected with  o
       -p  or  for  every  task  managed  by  the Linux kernel if option -p ALL has been used
       selecting any tasks is equivalent to specifying -p ALL but only active  tasks  (tasks
       non-zero statistics values) will appear in the report.

       The  pidstat command can also be used for monitoring the child processes of selected t
       Read about option -T below.

       The interval parameter specifies the amount of time in  seconds  between  each  report
       value of 0 (or no parameters at all) indicates that tasks statistics are to be reporte
       the time since system startup (boot).  The count parameter can be specified in  conjun
       with  the  interval parameter if this one is not set to zero. The value of count deter
       the number of reports generated at interval seconds apart. If  the  interval  paramete
       specified without the count parameter, the pidstat command generates reports continuou

       You  can select information about specific task activities using flags.  Not specifyin
       flags selects only CPU activity.

OPTIONS
       -C comm
              Display only tasks whose command name includes the string comm.  This string ca
              a regular expression.

       -d     Report  I/O statistics (kernels 2.6.20 and later only).  The following values m
              displayed:

              UID
                     The real user identification number of the task being monitored.

              USER
                     The name of the real user owning the task being monitored.

              PID
                     The identification number of the task being monitored.

              kB_rd/s
                     Number of kilobytes the task has caused to be read from disk per second.

              kB_wr/s
```

Used for monitoring of individual tasks currently being managed by the kernel.

Usage:

Memory utilization, page faults, stack utilization, CPU utilization, task switching,..

-d Report I/O statistics

-r Report page faults and memory utilization

-s Report stack utilization

-u Report CPU utilization

-w Report task switching activity

Sysstat family

iostat – Report CPU statics and input/output statistics for devices, partitions and network filesystems (NFS), eg monitoring IO on your dedicated disk.

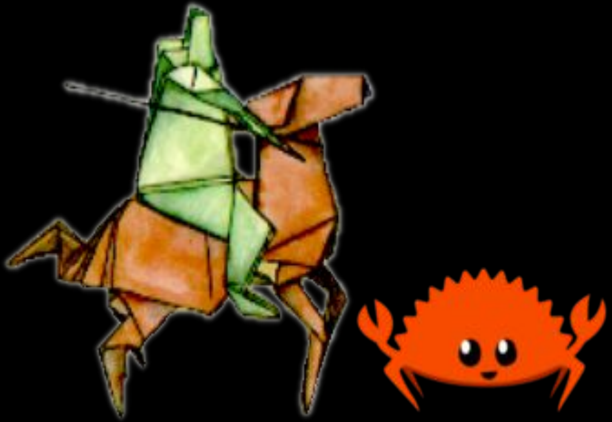
mpstat – Report processors related statistics, use eg for monitoring of pinned processes.

sar – Collect, report, or save system activity information.

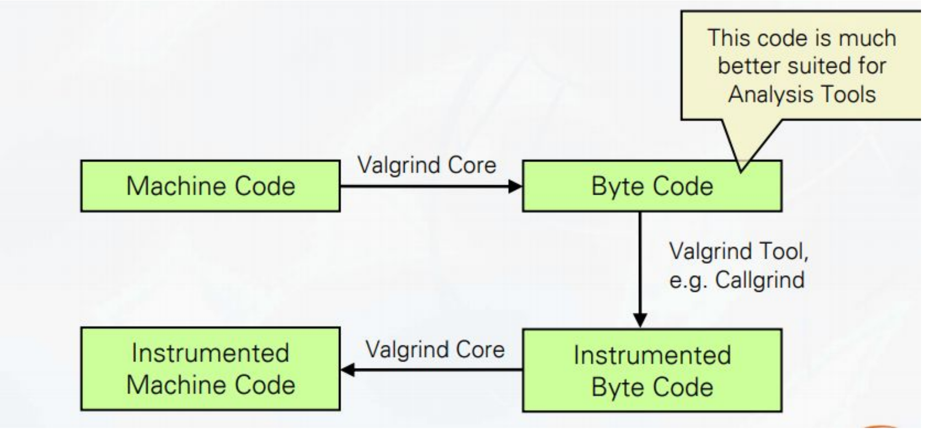
…

https://github.com/sysstat/sysstat

valgrind tools

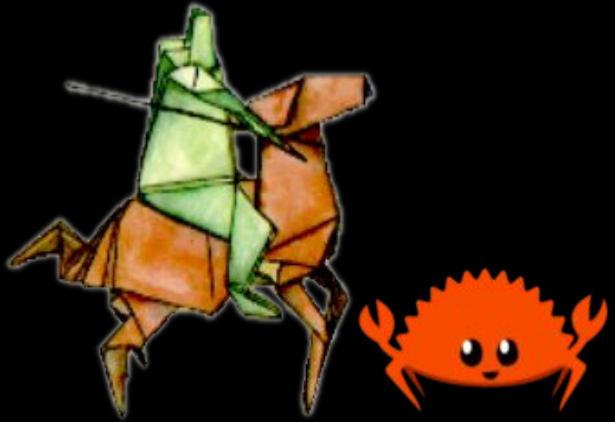


- Suite of simulation-based debugging and profiling tools
- program is run on a synthetic CPU provided by the Valgrind core
- Tools are adding own instrumentation code to binary to be handled by Valgrind’s core (Valgrind Tool = Valgrind Core + Tool Plugin)
- Large overhead (~10-100x slower)


This code is much better suited for Analysis Tools
Machine Code
Valgrind Core
Byte Code
Valgrind Tool, e.g. Callgrind
Instrumented Machine Code
Valgrind Core
Instrumented Byte Code

http://valgrind.org/docs/manual/manual.html
http://wwwpub.zih.tu-dresden.de/~mlieber/practical_performance/04_callgrind.pdf

valgrind tools



memcheck – memory error detection

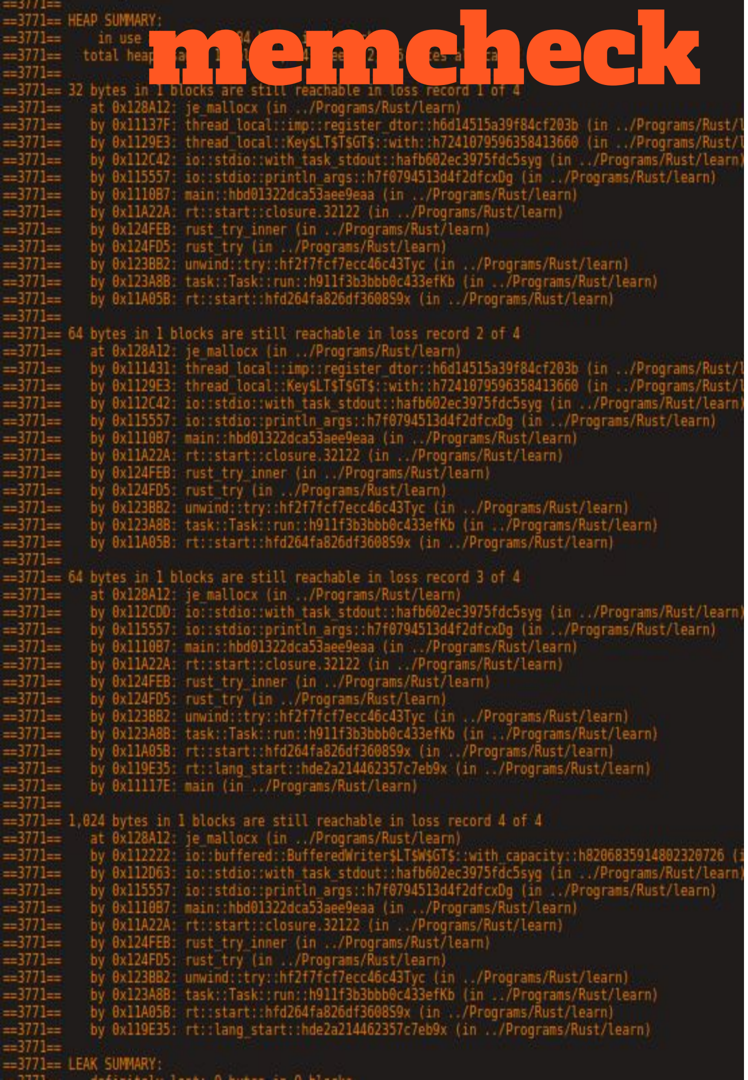
cachegrind – cache and branch prediction profiler

callgrind – a callgraph generating cache and branch prediction profiler

massif – heap profiler

helgrind, DRD – thread error detectors (experimental: stack/global array overrun detector, heap profiler – DHAT, that examines how heap blocks are used,..)

memcheck



```
==3771==
==3771== HEAP SUMMARY:
==3771==     in use [illegible]
==3771==   total heap [illegible]
==3771==
==3771== 32 bytes in 1 blocks are still reachable in loss record 1 of 4
==3771==    at 0x128A12: je_mallocx (in ../Programs/Rust/learn)
==3771==    by 0x11137F: thread_local::imp::register_dtor::h6d14515a39f84cf203b (in ../Programs/Rust/l
==3771==    by 0x1129E3: thread_local::Key$LT$T$GT$::with::h7241079596358413660 (in ../Programs/Rust/l
==3771==    by 0x112C42: io::stdio::with_task_stdout::hafb602ec3975fdc5syg (in ../Programs/Rust/learn)
==3771==    by 0x115557: io::stdio::println_args::h7f0794513d4f2dfcxDg (in ../Programs/Rust/learn)
==3771==    by 0x1110B7: main::hbd01322dca53aee9eaa (in ../Programs/Rust/learn)
==3771==    by 0x11A22A: rt::start::closure.32122 (in ../Programs/Rust/learn)
==3771==    by 0x124FEB: rust_try_inner (in ../Programs/Rust/learn)
==3771==    by 0x124FD5: rust_try (in ../Programs/Rust/learn)
==3771==    by 0x123BB2: unwind::try::hf2f7fcf7ecc46c43Tyc (in ../Programs/Rust/learn)
==3771==    by 0x123A8B: task::Task::run::h911f3b3bbb0c433efKb (in ../Programs/Rust/learn)
==3771==    by 0x11A05B: rt::start::hfd264fa826df3608S9x (in ../Programs/Rust/learn)
==3771==
==3771== 64 bytes in 1 blocks are still reachable in loss record 2 of 4
==3771==    at 0x128A12: je_mallocx (in ../Programs/Rust/learn)
==3771==    by 0x111431: thread_local::imp::register_dtor::h6d14515a39f84cf203b (in ../Programs/Rust/l
==3771==    by 0x1129E3: thread_local::Key$LT$T$GT$::with::h7241079596358413660 (in ../Programs/Rust/l
==3771==    by 0x112C42: io::stdio::with_task_stdout::hafb602ec3975fdc5syg (in ../Programs/Rust/learn)
==3771==    by 0x115557: io::stdio::println_args::h7f0794513d4f2dfcxDg (in ../Programs/Rust/learn)
==3771==    by 0x1110B7: main::hbd01322dca53aee9eaa (in ../Programs/Rust/learn)
==3771==    by 0x11A22A: rt::start::closure.32122 (in ../Programs/Rust/learn)
==3771==    by 0x124FEB: rust_try_inner (in ../Programs/Rust/learn)
==3771==    by 0x124FD5: rust_try (in ../Programs/Rust/learn)
==3771==    by 0x123BB2: unwind::try::hf2f7fcf7ecc46c43Tyc (in ../Programs/Rust/learn)
==3771==    by 0x123A8B: task::Task::run::h911f3b3bbb0c433efKb (in ../Programs/Rust/learn)
==3771==    by 0x11A05B: rt::start::hfd264fa826df3608S9x (in ../Programs/Rust/learn)
==3771==
==3771== 64 bytes in 1 blocks are still reachable in loss record 3 of 4
==3771==    at 0x128A12: je_mallocx (in ../Programs/Rust/learn)
==3771==    by 0x112CDD: io::stdio::with_task_stdout::hafb602ec3975fdc5syg (in ../Programs/Rust/learn)
==3771==    by 0x115557: io::stdio::println_args::h7f0794513d4f2dfcxDg (in ../Programs/Rust/learn)
==3771==    by 0x1110B7: main::hbd01322dca53aee9eaa (in ../Programs/Rust/learn)
==3771==    by 0x11A22A: rt::start::closure.32122 (in ../Programs/Rust/learn)
==3771==    by 0x124FEB: rust_try_inner (in ../Programs/Rust/learn)
==3771==    by 0x124FD5: rust_try (in ../Programs/Rust/learn)
==3771==    by 0x123BB2: unwind::try::hf2f7fcf7ecc46c43Tyc (in ../Programs/Rust/learn)
==3771==    by 0x123A8B: task::Task::run::h911f3b3bbb0c433efKb (in ../Programs/Rust/learn)
==3771==    by 0x11A05B: rt::start::hfd264fa826df3608S9x (in ../Programs/Rust/learn)
==3771==    by 0x119E35: rt::lang_start::hde2a214462357c7eb9x (in ../Programs/Rust/learn)
==3771==    by 0x11117E: main (in ../Programs/Rust/learn)
==3771==
==3771== 1,024 bytes in 1 blocks are still reachable in loss record 4 of 4
==3771==    at 0x128A12: je_mallocx (in ../Programs/Rust/learn)
==3771==    by 0x112222: io::buffered::BufferedWriter$LT$W$GT$::with_capacity::h8206835914802320726 (i
==3771==    by 0x112D63: io::stdio::with_task_stdout::hafb602ec3975fdc5syg (in ../Programs/Rust/learn)
==3771==    by 0x115557: io::stdio::println_args::h7f0794513d4f2dfcxDg (in ../Programs/Rust/learn)
==3771==    by 0x1110B7: main::hbd01322dca53aee9eaa (in ../Programs/Rust/learn)
==3771==    by 0x11A22A: rt::start::closure.32122 (in ../Programs/Rust/learn)
==3771==    by 0x124FEB: rust_try_inner (in ../Programs/Rust/learn)
==3771==    by 0x124FD5: rust_try (in ../Programs/Rust/learn)
==3771==    by 0x123BB2: unwind::try::hf2f7fcf7ecc46c43Tyc (in ../Programs/Rust/learn)
==3771==    by 0x123A8B: task::Task::run::h911f3b3bbb0c433efKb (in ../Programs/Rust/learn)
==3771==    by 0x11A05B: rt::start::hfd264fa826df3608S9x (in ../Programs/Rust/learn)
==3771==    by 0x119E35: rt::lang_start::hde2a214462357c7eb9x (in ../Programs/Rust/learn)
==3771==
==3771== LEAK SUMMARY:
==3771==    definitely lost: 0 bytes in 0 blocks
```

Memory related errors

- not so common in Rust due to RAII ??
- !! jemalloc's support for Valgrind controversy, use system allocator

Compile with debug symbols and no optimizations:

(runs 10-50 times slower than natively)

`valgrind -v --leak-check=full ./prog args..`

`--track-fds` a list of open file descriptors on exit or on request

`--trace-children` a list of open file descriptors on exit or on request

`--xtree-memory=full` Produces execution tree detailing which piece of code is responsible for heap memory usage; display via Kcachgrind

```
valgrind --tool=memcheck --leak-check=full --track-fds=yes
--track-origins=yes ./prog
```

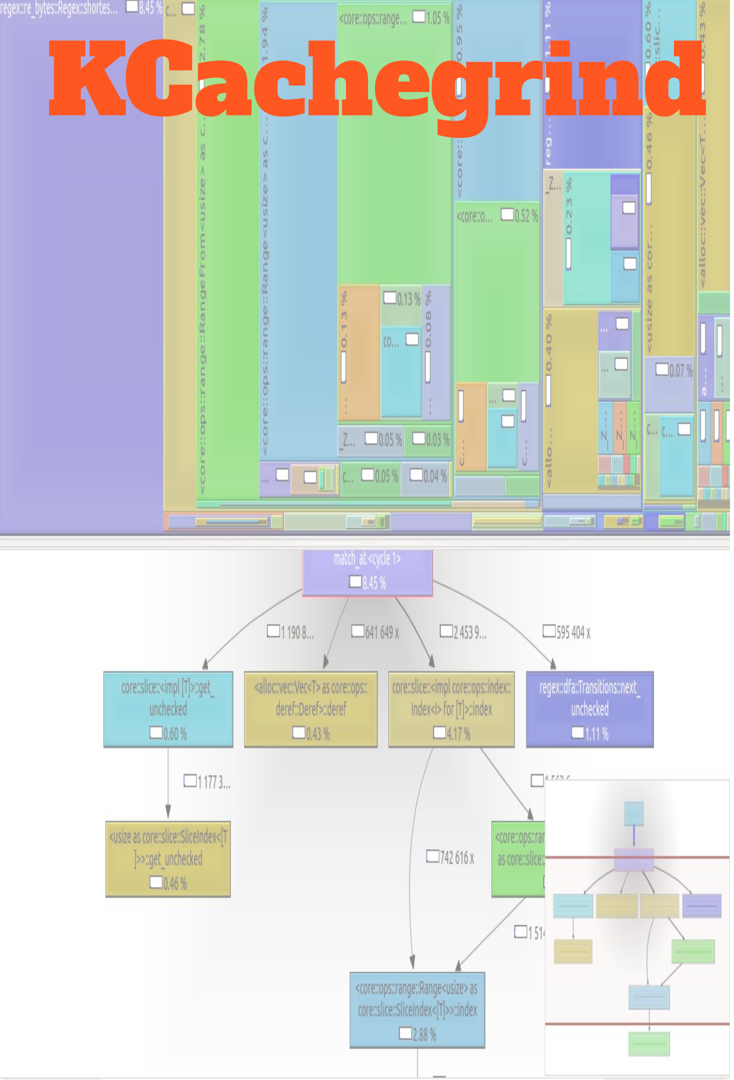
Types of error messages:

- Definitely lost – memory leaked
- Probably lost – memory leaking unless you do magic with pointers
- Uninitialised values – eg if conditional jump/move depends on uninitialised value

For full list: http://valgrind.org/docs/manual/mc-manual.html#mc-manual.errormsgs

Projects memcheck-free of errors (OpenOffice, Firefox,..)

KCachegrind


regex::re_bytes::Regex::shortes...
8.45 %
<core::ops::range...
1.05 %
<core::ops::range::RangeFrom<usize> as c...
2.78 %
<core::ops::range::Range<usize> as c...
1.94 %
0.13 %
0.08 %
0.05 %
0.03 %
0.04 %
0.52 %
0.40 %
0.23 %
0.07 %
match_at <cycle 1>
8.45 %
1 190 8...
641 649 x
2 453 9...
595 404 x
core::slice::<impl [T]>::get_unchecked
0.60 %
<alloc::vec::Vec<T> as core::ops::deref::Deref>::deref
0.43 %
core::slice::<impl core::ops::index::Index<I> for [T]>::index
4.17 %
regex::dfa::Transitions::next_unchecked
1.11 %
1 177 3...
<usize as core::slice::SliceIndex<[T]>>::get_unchecked
0.46 %
742 616 x
<core::ops::range::Range<usize> as core::slice::SliceIndex<[T]>>::index
2.88 %

Visualization tool for Valgrind’s memcheck, cachegrind, callgrind.

http://kcachegrind.sourceforge.net/html/Home.html

rust -san

rustc experimentally supported sanitizer

Compile with RUSTFLAGS set with -Z sanitizer flag

(runs ~2x slower than natively)

```
    RUSTFLAGS=”-Z sanitizer=$SAN” cargo run --target
x86_64-unknown-linux-gnu prog args
```

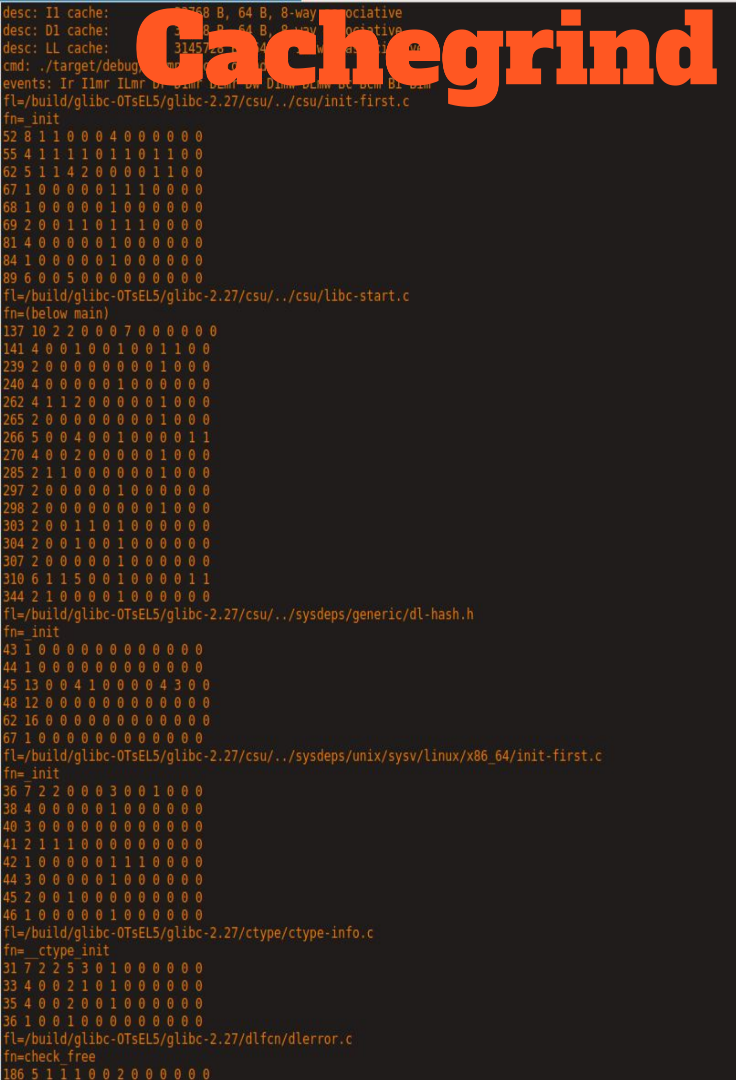
Can run with **$SAN**:

- **address** - detects out of bounds access and use of freed memory
- **leak** - detects memory leaks
- **memory** - reads on uninitialized memory, memory leaks
- **thread**

https://github.com/japaric/rust-san

https://users.rust-lang.org/t/howto-sanitize-your-rust-code/9378

Cachegrind



```
desc: I1 cache:         32768 B, 64 B, 8-way associative
desc: D1 cache:         32768 B, 64 B, 8-way associative
desc: LL cache:         3145728 B, 64 B, [illegible]-way associative
cmd: ./target/debug/[illegible]
events: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw Bc Bcm Bi Bim
fl=/build/glibc-OTsEL5/glibc-2.27/csu/../csu/init-first.c
fn=_init
52 8 1 1 0 0 0 4 0 0 0 0 0 0
55 4 1 1 1 1 0 1 1 0 1 1 0 0
62 5 1 1 4 2 0 0 0 0 1 1 0 0
67 1 0 0 0 0 0 1 1 1 0 0 0 0
68 1 0 0 0 0 0 1 0 0 0 0 0 0
69 2 0 0 1 1 0 1 1 1 0 0 0 0
81 4 0 0 0 0 0 1 0 0 0 0 0 0
84 1 0 0 0 0 0 1 0 0 0 0 0 0
89 6 0 0 5 0 0 0 0 0 0 0 0 0
fl=/build/glibc-OTsEL5/glibc-2.27/csu/../csu/libc-start.c
fn=(below main)
137 10 2 2 0 0 0 7 0 0 0 0 0 0
141 4 0 0 1 0 0 1 0 0 1 1 0 0
239 2 0 0 0 0 0 0 0 0 1 0 0 0
240 4 0 0 0 0 0 1 0 0 0 0 0 0
262 4 1 1 2 0 0 0 0 0 1 0 0 0
265 2 0 0 0 0 0 0 0 0 1 0 0 0
266 5 0 0 4 0 0 1 0 0 0 0 1 1
270 4 0 0 2 0 0 0 0 0 1 0 0 0
285 2 1 1 0 0 0 0 0 0 1 0 0 0
297 2 0 0 0 0 0 1 0 0 0 0 0 0
298 2 0 0 0 0 0 0 0 0 1 0 0 0
303 2 0 0 1 1 0 1 0 0 0 0 0 0
304 2 0 0 1 0 0 1 0 0 0 0 0 0
307 2 0 0 0 0 0 1 0 0 0 0 0 0
310 6 1 1 5 0 0 1 0 0 0 0 1 1
344 2 1 0 0 0 0 1 0 0 0 0 0 0
fl=/build/glibc-OTsEL5/glibc-2.27/csu/../sysdeps/generic/dl-hash.h
fn=_init
43 1 0 0 0 0 0 0 0 0 0 0 0 0
44 1 0 0 0 0 0 0 0 0 0 0 0 0
45 13 0 0 4 1 0 0 0 0 4 3 0 0
48 12 0 0 0 0 0 0 0 0 0 0 0 0
62 16 0 0 0 0 0 0 0 0 0 0 0 0
67 1 0 0 0 0 0 0 0 0 0 0 0 0
fl=/build/glibc-OTsEL5/glibc-2.27/csu/../sysdeps/unix/sysv/linux/x86_64/init-first.c
fn=_init
36 7 2 2 0 0 0 3 0 0 1 0 0 0
38 4 0 0 0 0 0 1 0 0 0 0 0 0
40 3 0 0 0 0 0 0 0 0 0 0 0 0
41 2 1 1 1 0 0 0 0 0 0 0 0 0
42 1 0 0 0 0 0 1 1 1 0 0 0 0
44 3 0 0 0 0 0 1 0 0 0 0 0 0
45 2 0 0 1 0 0 0 0 0 0 0 0 0
46 1 0 0 0 0 0 1 0 0 0 0 0 0
fl=/build/glibc-OTsEL5/glibc-2.27/ctype/ctype-info.c
fn=__ctype_init
31 7 2 2 5 3 0 1 0 0 0 0 0 0
33 4 0 0 2 1 0 1 0 0 0 0 0 0
35 4 0 0 2 0 0 1 0 0 0 0 0 0
36 1 0 0 1 0 0 0 0 0 0 0 0 0
fl=/build/glibc-OTsEL5/glibc-2.27/dlfcn/dlerror.c
fn=check_free
186 5 1 1 1 0 0 2 0 0 0 0 0 0
```

Cache and branch prediction profiler.

Compile with debug symbols and optimizations in:

```
valgrind --tool=cachegrind ./prog args..
```

`--branch-sim=yes` branch prediction statistics

`--trace-children` trace children processes

Result displays, eg.:

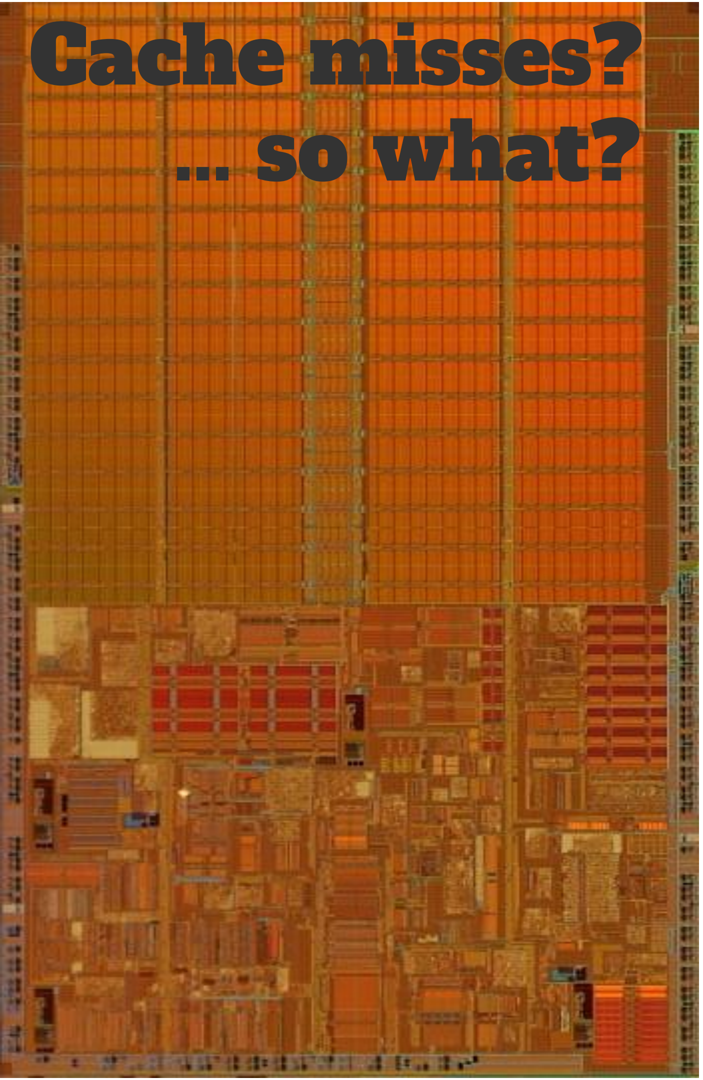
- Unified first level instructions and data caches (I1 and D1)
- Unified second (last) level cache information (I2 or Il)
- Branch misprediction
- !! cache misses visible at the instruction level only, e.g. those arising from TLB misses, or speculative execution will not be included

Cachegrind gathers:

- I cache reads (Ir, which equals the number of instructions executed), I1 cache read misses (I1mr) and LL cache instruction read misses (ILmr).
- D cache reads (Dr, which equals the number of memory reads), D1 cache read misses (D1mr), and LL cache data read misses (DLmr).
- D cache writes (Dw, which equals the number of memory writes), D1 cache write misses (D1mw), and LL cache data write misses (DLmw).
- Conditional branches executed (Bc) and conditional branches mispredicted (Bcm).
- Indirect branches executed (Bi) and indirect branches mispredicted (Bim).

http://valgrind.org/docs/manual/cg-manual.html

Cache misses?
… so what?

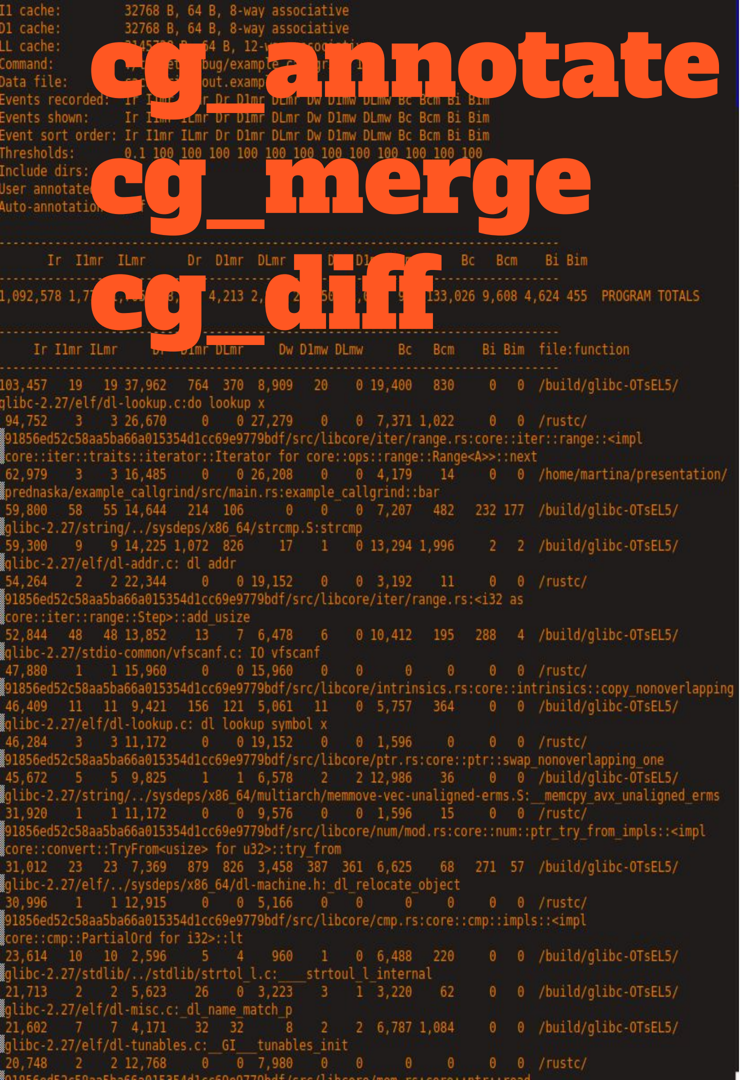


Price for cache misses on modern machines:

- L1 miss ~10 cycles
- L2 miss ~15 cycles
- LL miss ~200 cycles
- Mispredicted branch ~10-30 cycles

Detailed branch profiling can help understand how program interacts with the machine and how to make it faster.

Further reading on caches:
https://www.extremetech.com/extreme/188776-how-l1-and-l2-cpu-caches-work-and-why-theyre-an-essential-part-of-modern-chips



cg_annotate
cg_merge
cg_diff

```
I1 cache:         32768 B, 64 B, 8-way associative
D1 cache:         32768 B, 64 B, 8-way associative
LL cache:         3145728 B, 64 B, 12-w
Command:                  bug/example
Data file:                out.examp
Events recorded:  Ir I        r Dr D1mr DLmr Dw D1mw DLmw Bc Bcm Bi Bim
Events shown:     Ir I        Lmr Dr D1mr DLmr Dw D1mw DLmw Bc Bcm Bi Bim
Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw Bc Bcm Bi Bim
Thresholds:       0.1 100 100 100 100 100 100 100 100 100 100 100 100
Include dirs:
User annotate
Auto-annotation

--------------------------------------------------------------------------------
     Ir  I1mr  ILmr      Dr  D1mr  DLmr     D    D1               Bc    Bcm    Bi Bim
--------------------------------------------------------------------------------
1,092,578 1,77            4,213 2,                            133,026 9,608 4,624 455  PROGRAM TOTALS

--------------------------------------------------------------------------------
    Ir I1mr ILmr         D1mr DLmr     Dw D1mw DLmw     Bc    Bcm    Bi Bim  file:function
--------------------------------------------------------------------------------
103,457   19   19 37,962   764   370  8,909   20    0 19,400   830     0   0  /build/glibc-OTsEL5/
glibc-2.27/elf/dl-lookup.c:do lookup x
 94,752    3    3 26,670     0     0 27,279    0    0  7,371 1,022     0   0  /rustc/
91856ed52c58aa5ba66a015354d1cc69e9779bdf/src/libcore/iter/range.rs:core::iter::range::<impl
core::iter::traits::iterator::Iterator for core::ops::range::Range<A>>::next
 62,979    3    3 16,485     0     0 26,208    0    0  4,179    14     0   0  /home/martina/presentation/
prednaska/example_callgrind/src/main.rs:example_callgrind::bar
 59,800   58   55 14,644   214   106      0    0    0  7,207   482   232 177  /build/glibc-OTsEL5/
glibc-2.27/string/../sysdeps/x86_64/strcmp.S:strcmp
 59,300    9    9 14,225 1,072   826     17    1    0 13,294 1,996     2   2  /build/glibc-OTsEL5/
glibc-2.27/elf/dl-addr.c: dl addr
 54,264    2    2 22,344     0     0 19,152    0    0  3,192    11     0   0  /rustc/
91856ed52c58aa5ba66a015354d1cc69e9779bdf/src/libcore/iter/range.rs:<i32 as
core::iter::range::Step>::add_usize
 52,844   48   48 13,852    13     7  6,478    6    0 10,412   195   288   4  /build/glibc-OTsEL5/
glibc-2.27/stdio-common/vfscanf.c: IO vfscanf
 47,880    1    1 15,960     0     0 15,960    0    0      0     0     0   0  /rustc/
91856ed52c58aa5ba66a015354d1cc69e9779bdf/src/libcore/intrinsics.rs:core::intrinsics::copy_nonoverlapping
 46,409   11   11  9,421   156   121  5,061   11    0  5,757   364     0   0  /build/glibc-OTsEL5/
glibc-2.27/elf/dl-lookup.c: dl lookup symbol x
 46,284    3    3 11,172     0     0 19,152    0    0  1,596     0     0   0  /rustc/
91856ed52c58aa5ba66a015354d1cc69e9779bdf/src/libcore/ptr.rs:core::ptr::swap_nonoverlapping_one
 45,672    5    5  9,825     1     1  6,578    2    2 12,986    36     0   0  /build/glibc-OTsEL5/
glibc-2.27/string/../sysdeps/x86_64/multiarch/memmove-vec-unaligned-erms.S:__memcpy_avx_unaligned_erms
 31,920    1    1 11,172     0     0  9,576    0    0  1,596    15     0   0  /rustc/
91856ed52c58aa5ba66a015354d1cc69e9779bdf/src/libcore/num/mod.rs:core::num::ptr_try_from_impls::<impl
core::convert::TryFrom<usize> for u32>::try_from
 31,012   23   23  7,369   879   826  3,458  387  361  6,625    68   271  57  /build/glibc-OTsEL5/
glibc-2.27/elf/../sysdeps/x86_64/dl-machine.h:_dl_relocate_object
 30,996    1    1 12,915     0     0  5,166    0    0      0     0     0   0  /rustc/
91856ed52c58aa5ba66a015354d1cc69e9779bdf/src/libcore/cmp.rs:core::cmp::impls::<impl
core::cmp::PartialOrd for i32>::lt
 23,614   10   10  2,596     5     4    960    1    0  6,488   220     0   0  /build/glibc-OTsEL5/
glibc-2.27/stdlib/../stdlib/strtol_l.c:____strtoul_l_internal
 21,713    2    2  5,623    26     0  3,223    3    1  3,220    62     0   0  /build/glibc-OTsEL5/
glibc-2.27/elf/dl-misc.c:_dl_name_match_p
 21,602    7    7  4,171    32    32      8    2    2  6,787 1,084     0   0  /build/glibc-OTsEL5/
glibc-2.27/elf/dl-tunables.c:__GI___tunables_init
 20,748    2    2 12,768     0     0  7,980    0    0      0     0     0   0  /rustc/
```

cg_annotate:

Display results from Cachegrind files.

- global level, with configurable threshold
- line by line counts for specific file

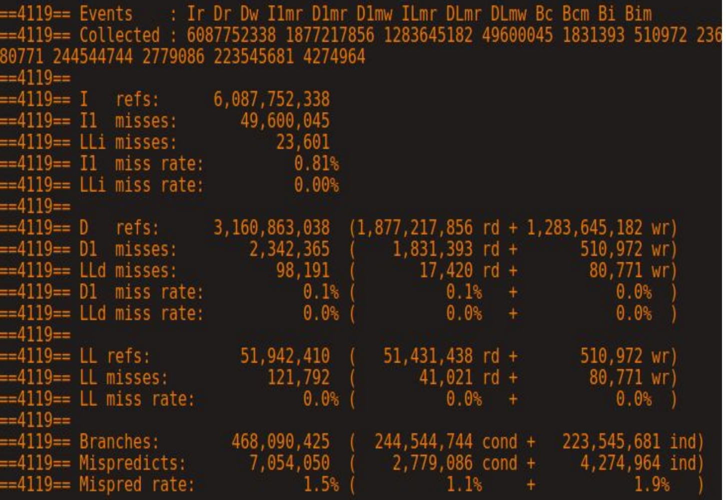
cg_merge:

Merge multiple profile files; might be useful to aggregate cost over multiple runs of the same program.

cg_diff:

Find differences between two profiles; measure how change to a program affected its performance.

callgrind



```
==4119== Events    : Ir Dr Dw I1mr D1mr D1mw ILmr DLmr DLmw Bc Bcm Bi Bim
==4119== Collected : 6087752338 1877217856 1283645182 49600045 1831393 510972 236
80771 244544744 2779086 223545681 4274964
==4119==
==4119== I   refs:      6,087,752,338
==4119== I1  misses:       49,600,045
==4119== LLi misses:           23,601
==4119== I1  miss rate:          0.81%
==4119== LLi miss rate:          0.00%
==4119==
==4119== D   refs:      3,160,863,038  (1,877,217,856 rd + 1,283,645,182 wr)
==4119== D1  misses:        2,342,365  (    1,831,393 rd +       510,972 wr)
==4119== LLd misses:           98,191  (       17,420 rd +        80,771 wr)
==4119== D1  miss rate:           0.1% (          0.1%   +           0.0%  )
==4119== LLd miss rate:           0.0% (          0.0%   +           0.0%  )
==4119==
==4119== LL refs:          51,942,410  (   51,431,438 rd +       510,972 wr)
==4119== LL misses:           121,792  (       41,021 rd +        80,771 wr)
==4119== LL miss rate:            0.0% (          0.0%   +           0.0%  )
==4119==
==4119== Branches:        468,090,425  (  244,544,744 cond +   223,545,681 ind)
==4119== Mispredicts:       7,054,050  (    2,779,086 cond +     4,274,964 ind)
==4119== Mispred rate:            1.5% (          1.1%     +           1.9%   )
```

Records the call history among functions in a program’s run as a call-graph.

- Callgrind shows inclusive cost, where the cost of each function includes the cost of all functions it called directly or indirectly
- Built on top of cachegrind
- Platform dependant (does not work on eg. ARM)

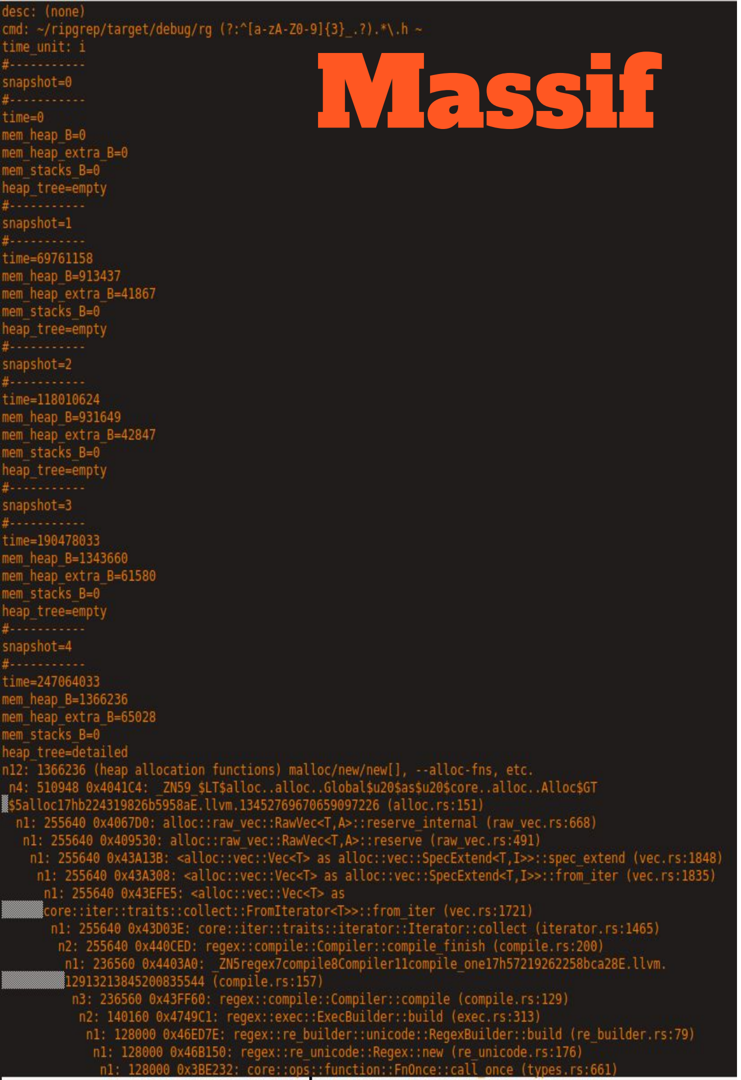
Compile with debug symbols and optimizations on:

```
valgrind --tool=callgrind [callgrind options] ./prog args
```

Textual output via callgrind_annotate, callgrind_control.

!! Only measures CPU time, so sleeping times are not included. This makes it unsuitable for programs that wait a significant amount of time for network or disk operations to complete.

Massif



```
desc: (none)
cmd: ~/ripgrep/target/debug/rg (?:^[a-zA-Z0-9]{3}_.?).*\.h ~
time_unit: i
#-----------
snapshot=0
#-----------
time=0
mem_heap_B=0
mem_heap_extra_B=0
mem_stacks_B=0
heap_tree=empty
#-----------
snapshot=1
#-----------
time=69761158
mem_heap_B=913437
mem_heap_extra_B=41867
mem_stacks_B=0
heap_tree=empty
#-----------
snapshot=2
#-----------
time=118010624
mem_heap_B=931649
mem_heap_extra_B=42847
mem_stacks_B=0
heap_tree=empty
#-----------
snapshot=3
#-----------
time=190478033
mem_heap_B=1343660
mem_heap_extra_B=61580
mem_stacks_B=0
heap_tree=empty
#-----------
snapshot=4
#-----------
time=247064033
mem_heap_B=1366236
mem_heap_extra_B=65028
mem_stacks_B=0
heap_tree=detailed
n12: 1366236 (heap allocation functions) malloc/new/new[], --alloc-fns, etc.
 n4: 510948 0x4041C4: _ZN59_$LT$alloc..alloc..Global$u20$as$u20$core..alloc..Alloc$GT
 $5alloc17hb224319826b5958aE.llvm.13452769670659097226 (alloc.rs:151)
  n1: 255640 0x4067D0: alloc::raw_vec::RawVec<T,A>::reserve_internal (raw_vec.rs:668)
   n1: 255640 0x409530: alloc::raw_vec::RawVec<T,A>::reserve (raw_vec.rs:491)
    n1: 255640 0x43A13B: <alloc::vec::Vec<T> as alloc::vec::SpecExtend<T,I>>::spec_extend (vec.rs:1848)
     n1: 255640 0x43A308: <alloc::vec::Vec<T> as alloc::vec::SpecExtend<T,I>>::from_iter (vec.rs:1835)
      n1: 255640 0x43EFE5: <alloc::vec::Vec<T> as
      core::iter::traits::collect::FromIterator<T>>::from_iter (vec.rs:1721)
       n1: 255640 0x43D03E: core::iter::traits::iterator::Iterator::collect (iterator.rs:1465)
        n2: 255640 0x440CED: regex::compile::Compiler::compile_finish (compile.rs:200)
         n1: 236560 0x4403A0: _ZN5regex7compile8Compiler11compile_one17h57219262258bca28E.llvm.
         12913213845200835544 (compile.rs:157)
          n3: 236560 0x43FF60: regex::compile::Compiler::compile (compile.rs:129)
           n2: 140160 0x4749C1: regex::exec::ExecBuilder::build (exec.rs:313)
            n1: 128000 0x46ED7E: regex::re_builder::unicode::RegexBuilder::build (re_builder.rs:79)
             n1: 128000 0x46B150: regex::re_unicode::Regex::new (re_unicode.rs:176)
              n1: 128000 0x3BE232: core::ops::function::FnOnce::call_once (types.rs:661)
```

Measures heap memory usage (optional stack usage). Display how much and where the memory was allocated.

Yes: malloc,calloc, realloc, memalign, new, new[]
No: memory allocated with low-level system calls like mmap, mremap, brk (pages-as-heap=yes, includes also stack)

Compile with debug symbols, optimisations do not matter

```
valgrind --tool=massif --threshold=0.01 ./prog args
```

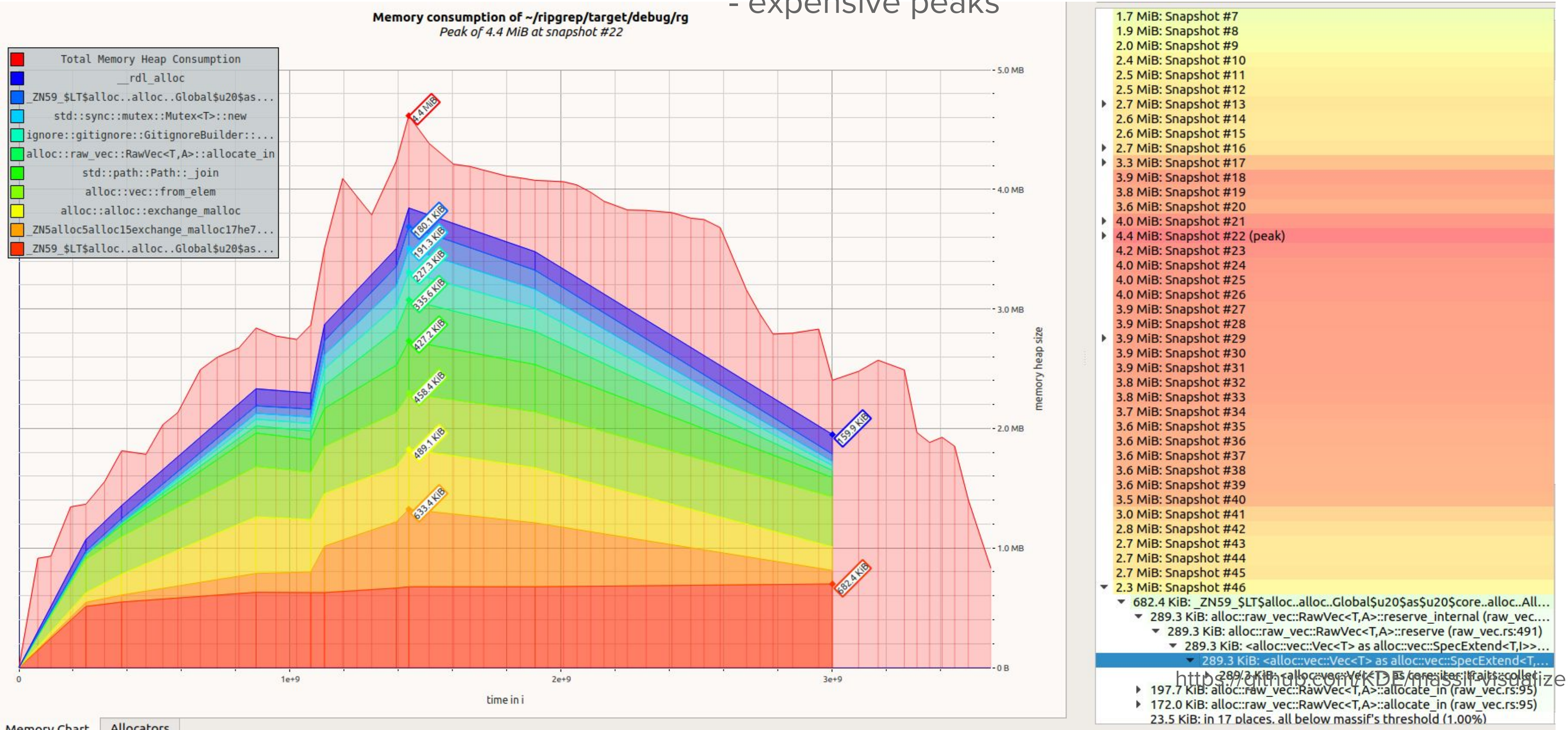
Benefits:

- Reduce the amount of memory program uses ➔ speed up program, helps avoid caching and prevents memory swapping
- Detection of memory leaks

massif_visualizer

Tool that visualizes massif data

- locations that significantly contribute to overall memory consumption
- memory leaks
- expensive peaks


Memory consumption of ~/ripgrep/target/debug/rg
Peak of 4.4 MiB at snapshot #22
🟥 Total Memory Heap Consumption
🟦 __rdl_alloc
🟦 _ZN59_$LT$alloc..alloc..Global$u20$as...
🟦 std::sync::mutex::Mutex<T>::new
🟩 ignore::gitignore::GitignoreBuilder::...
🟩 alloc::raw_vec::RawVec<T,A>::allocate_in
🟩 std::path::Path::_join
🟩 alloc::vec::from_elem
🟨 alloc::alloc::exchange_malloc
🟧 _ZN5alloc5alloc15exchange_malloc17he7...
🟥 _ZN59_$LT$alloc..alloc..Global$u20$as...
4.4 MiB
180.1 KiB
191.3 KiB
227.3 KiB
335.6 KiB
427.2 KiB
458.4 KiB
489.1 KiB
633.4 KiB
159.9 KiB
682.4 KiB
5.0 MB
4.0 MB
3.0 MB
2.0 MB
1.0 MB
0 B
memory heap size
0
1e+9
2e+9
3e+9
time in i
Memory Chart
Allocators

```
1.7 MiB: Snapshot #7
1.9 MiB: Snapshot #8
2.0 MiB: Snapshot #9
2.4 MiB: Snapshot #10
2.5 MiB: Snapshot #11
2.5 MiB: Snapshot #12
2.7 MiB: Snapshot #13
2.6 MiB: Snapshot #14
2.6 MiB: Snapshot #15
2.7 MiB: Snapshot #16
3.3 MiB: Snapshot #17
3.9 MiB: Snapshot #18
3.8 MiB: Snapshot #19
3.6 MiB: Snapshot #20
4.0 MiB: Snapshot #21
4.4 MiB: Snapshot #22 (peak)
4.2 MiB: Snapshot #23
4.0 MiB: Snapshot #24
4.0 MiB: Snapshot #25
4.0 MiB: Snapshot #26
3.9 MiB: Snapshot #27
3.9 MiB: Snapshot #28
3.9 MiB: Snapshot #29
3.9 MiB: Snapshot #30
3.9 MiB: Snapshot #31
3.8 MiB: Snapshot #32
3.8 MiB: Snapshot #33
3.7 MiB: Snapshot #34
3.6 MiB: Snapshot #35
3.6 MiB: Snapshot #36
3.6 MiB: Snapshot #37
3.6 MiB: Snapshot #38
3.6 MiB: Snapshot #39
3.5 MiB: Snapshot #40
3.0 MiB: Snapshot #41
2.8 MiB: Snapshot #42
2.7 MiB: Snapshot #43
2.7 MiB: Snapshot #44
2.7 MiB: Snapshot #45
2.3 MiB: Snapshot #46
  682.4 KiB: _ZN59_$LT$alloc..alloc..Global$u20$as$u20$core..alloc..All...
    289.3 KiB: alloc::raw_vec::RawVec<T,A>::reserve_internal (raw_vec....
      289.3 KiB: alloc::raw_vec::RawVec<T,A>::reserve (raw_vec.rs:491)
        289.3 KiB: <alloc::vec::Vec<T> as alloc::vec::SpecExtend<T,I>>...
          289.3 KiB: <alloc::vec::Vec<T> as alloc::vec::SpecExtend<T,...
            289.3 KiB: <alloc::vec::Vec<T> as core::iter::traits::collect...
    197.7 KiB: alloc::raw_vec::RawVec<T,A>::allocate_in (raw_vec.rs:95)
    172.0 KiB: alloc::raw_vec::RawVec<T,A>::allocate_in (raw_vec.rs:95)
    23.5 KiB: in 17 places, all below massif's threshold (1.00%)
```

https://github.com/KDE/massif-visualizer

Perf (perf_events)

```
==3771==
==3771== HEAP SUMMARY:
==3771==     in use at exit: 1,184 bytes in 4 blocks
==3771==   total heap usage: 18 allocs, 14 frees, 2,696 byte[illegible]ted
==3771==
==3771== 32 bytes in 1 blocks are still reachable in loss re[illegible] 1 o
==3771==    at 0x128A12: je_mallocx (in ../Programs/Rust/le[illegible]
==3771==    by 0x11137F: thread_local::imp::register_dtor::h6d14515a39f84cf203b (in ../Programs/Rust/l
==3771==    by 0x1129E3: thread_local::Key$LT$T$GT$::with::h7241079596358413660 (in ../Programs/Rust/l
==3771==    by 0x112C42: io::stdio::[illegible] task_stdout::hafb602ec3975fdc5syg (in ../Programs/Rus[illegible]earn)
==3771==    by 0x[illegible]ntln_args::[illegible]
==3771==    [illegible]dca53aee9[illegible]
==3771==    [illegible]sure.3212[illegible]
==3771==    [illegible]x124FEB: rust_try_inner (in ../Programs/Rust/learn)
==3771==    [illegible]x124FD5: rust_try (in ../Programs/Rust/learn)
==3771==    by 0x123BB2: unwind::try::hf2f7fcf7ecc46c43Tyc (in ../Programs/Rust/learn)
==3771==    by 0x123A8B: task::Task::run::h911f3b3bbb0c433efKb (in ../Programs/Rust/learn)
==3771==    by 0x11A05B: rt::start::hfd264fa826df3608S9x (in ../Programs/Rust/learn)
==3771==
==3771== 64 bytes in 1 blocks are still reachable in loss record 2 of 4
==3771==    at 0x128A12: je_mallocx (in ../Programs/Rust/learn)
==3771==    by 0x111431: thread_local::imp::register_dtor::h6d14515a39f84cf203b (in ../Programs/Rust/l
==3771==    by 0x1129E3: thread_local::Key$LT$T$GT$::with::h7241079596358413660 (in ../Programs/Rust/l
==3771==    by 0x112C42: io::stdio::with_task_stdout::hafb602ec3975fdc5syg (in ../Programs/Rust/learn)
==3771==    by 0x115557: io::stdio::println_args::h7f0794513d4f2dfcxDg (in ../Programs/Rust/learn)
==3771==    by 0x1110B7: main::hbd01322dca53aee9eaa (in ../Programs/Rust/learn)
==3771==    by 0x11A22A: rt::start::closure.32122 (in ../Programs/Rust/learn)
==3771==    by 0x124FEB: rust_try_inner (in ../Programs/Rust/learn)
==3771==    by 0x124FD5: rust_try (in ../Programs/Rust/learn)
==3771==    by 0x123BB2: unwind::try::hf2f7fcf7ecc46c43Tyc (in ../Programs/Rust/learn)
==3771==    by 0x123A8B: task::Task::run::h911f3b3bbb0c433efKb (in ../Programs/Rust/learn)
==3771==    by 0x11A05B: rt::start::hfd264fa826df3608S9x (in ../Programs/Rust/learn)
==3771==
==3771== 64 bytes in 1 blocks are still reachable in loss record 3 of 4
==3771==    at 0x128A12: je_mallocx (in ../Programs/Rust/learn)
==3771==    by 0x112CDD: io::stdio::with_task_stdout::hafb602ec3975fdc5syg (in ../Programs/Rust/learn)
==3771==    by 0x115557: io::stdio::println_args::h7f0794513d4f2dfcxDg (in ../Programs/Rust/learn)
==3771==    by 0x1110B7: main::hbd01322dca53aee9eaa (in ../Programs/Rust/learn)
==3771==    by 0x11A22A: rt::start::closure.32122 (in ../Programs/Rust/learn)
==3771==    by 0x124FEB: rust_try_inner (in ../Programs/Rust/learn)
==3771==    by 0x124FD5: rust_try (in ../Programs/Rust/learn)
==3771==    by 0x123BB2: unwind::try::hf2f7fcf7ecc46c43Tyc (in ../Programs/Rust/learn)
==3771==    by 0x123A8B: task::Task::run::h911f3b3bbb0c433efKb (in ../Programs/Rust/learn)
==3771==    by 0x11A05B: rt::start::hfd264fa826df3608S9x (in ../Programs/Rust/learn)
==3771==    by 0x119E35: rt::lang_start::hde2a214462357c7eb9x (in ../Programs/Rust/learn)
==3771==    by 0x11117E: main (in ../Programs/Rust/learn)
==3771==
==3771== 1,024 bytes in 1 blocks are still reachable in loss record 4 of 4
==3771==    at 0x128A12: je_mallocx (in ../Programs/Rust/learn)
==3771==    by 0x112222: io::buffered::BufferedWriter$LT$W$GT$::with_capacity::h8206835914802320726 (i
==3771==    by 0x112D63: io::stdio::with_task_stdout::hafb602ec3975fdc5syg (in ../Programs/Rust/learn)
==3771==    by 0x115557: io::stdio::println_args::h7f0794513d4f2dfcxDg (in ../Programs/Rust/learn)
==3771==    by 0x1110B7: main::hbd01322dca53aee9eaa (in ../Programs/Rust/learn)
==3771==    by 0x11A22A: rt::start::closure.32122 (in ../Programs/Rust/learn)
==3771==    by 0x124FEB: rust_try_inner (in ../Programs/Rust/learn)
==3771==    by 0x124FD5: rust_try (in ../Programs/Rust/learn)
==3771==    by 0x123BB2: unwind::try::hf2f7fcf7ecc46c43Tyc (in ../Programs/Rust/learn)
==3771==    by 0x123A8B: task::Task::run::h911f3b3bbb0c433efKb (in ../Programs/Rust/learn)
==3771==    by 0x11A05B: rt::start::hfd264fa826df3608S9x (in ../Programs/Rust/learn)
==3771==    by 0x119E35: rt::lang_start::hde2a214462357c7eb9x (in ../Programs/Rust/learn)
==3771==
==3771== LEAK SUMMARY:
```

Event oriented observability tool which can help you solve advanced performance and troubleshooting functions.

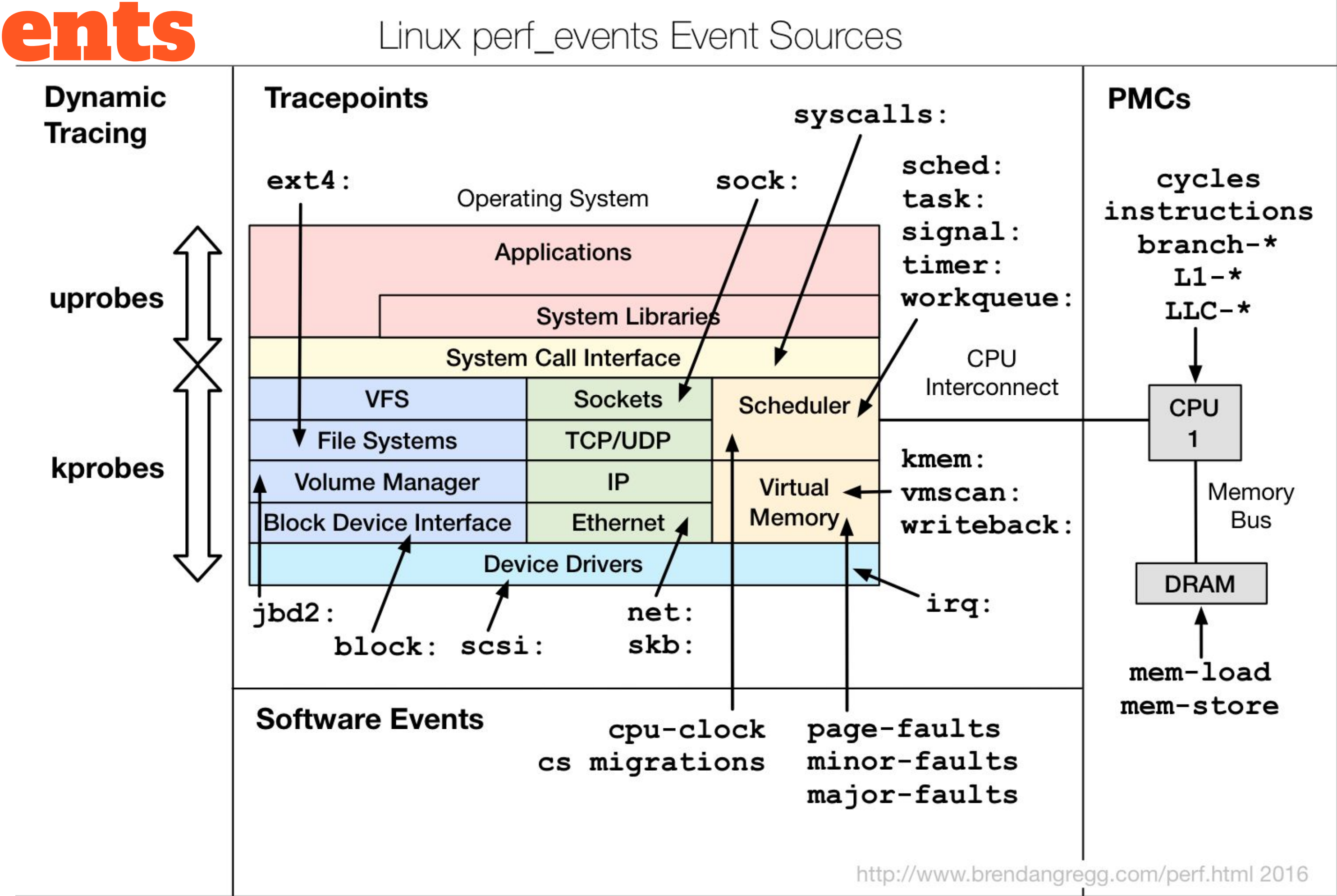
Compile with debug symbols and frame pointers (build with **-fno-omit-frame-pointers;** should be when debug on??).

For full list of all known events: (perf list)

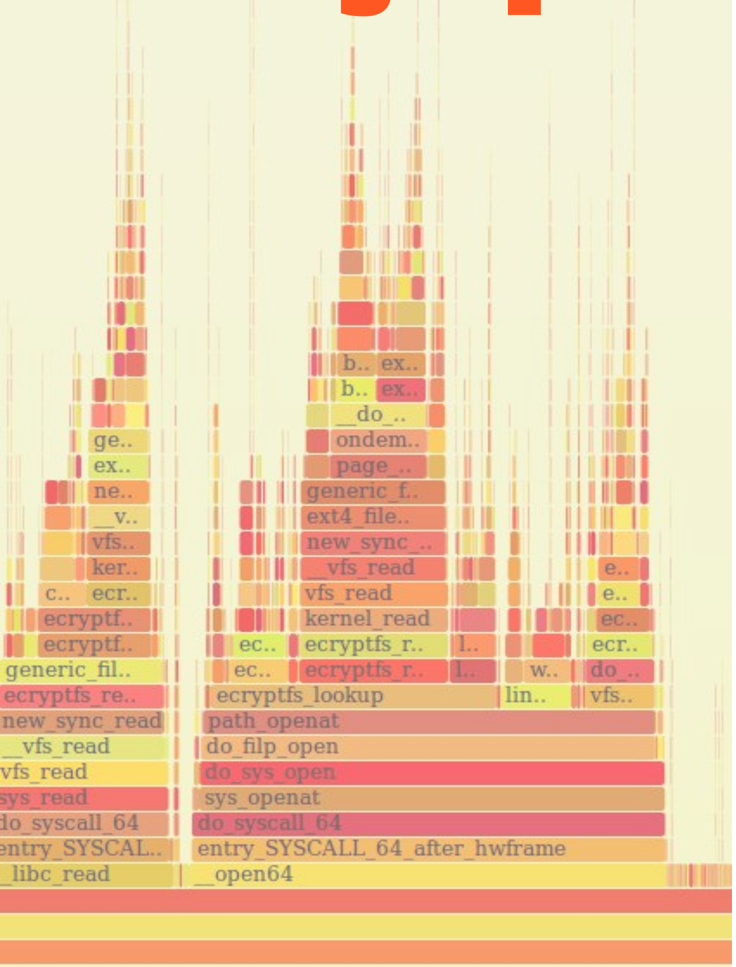
- Event counting (perf stat)
- Profiling (perf record or perf top)
- Static tracing (perf record)
- Dynamic tracing (perf probe)
- Reporting (perf report)
- Cache-2-cache and chacheline false sharing analysis (perf c2c)
- Kernel allocation analysis (perf kmem)
- Lock analysis (perf lock)
- Memory access analysis (perf mem)
- Kernel scheduler analysis (perf sched)

http://www.brendangregg.com/perf.html

Perf_events Events sources



flamegraph



Helps quantify code-paths and determine places with performance problems

Visualization of perf.data from perf report:

- X axis sample population
- Y axis stack depth
- Width of each function is relative to the number of samples.

1. Capture stack:

```
perf record -F 99 --call-graph=dwarf -p PID
perf script > out.perf
```

2. Use the stackcollapse programs to fold stack samples into single lines:

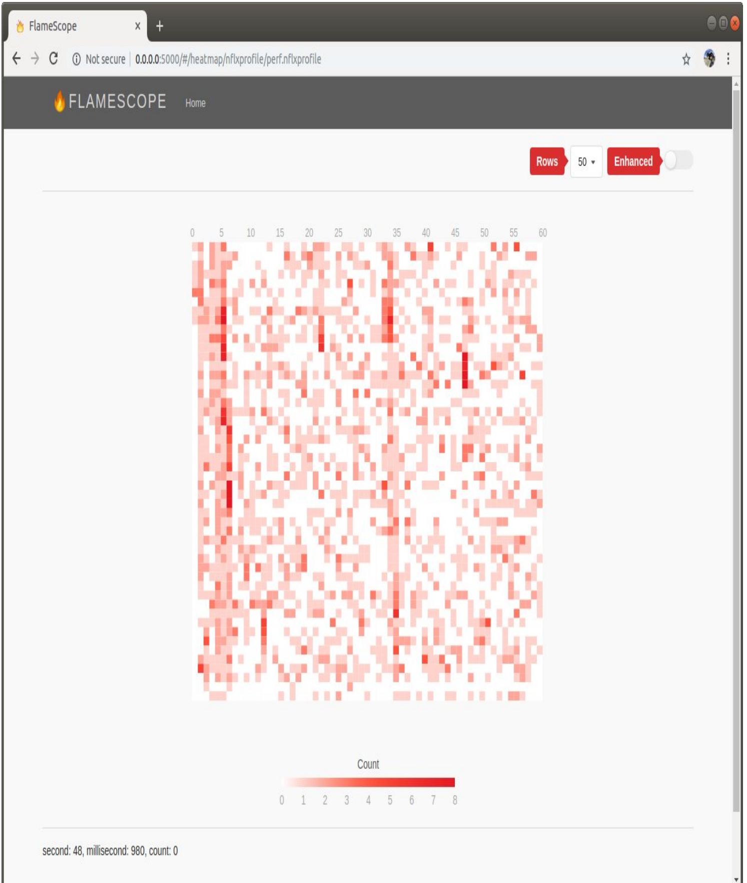
```
./stackcollapse-perf.pl out.perf > out.folded
```

3. Use flamegraph.pl to render a SVG.

```
./flamegraph.pl out.kern_folded > kernel.svg
```

http://www.brendangregg.com/flamegraphs.html

FlameScope



second: 48, millisecond: 980, count: 0

- Interactive visualization tool for different timeranges of FlameGraph in the form of heat map

- perf script's output that includes stack traces, including page faults, context switches, and other events.

```
sudo perf record -F _FREQUENCY_ -p PID
sudo perf script --header > flamescope_out_file
```

https://github.com/Netflix/flamescope
https://www.youtube.com/watch?v=cFuI8SAAvJg

Google perf tools gperftools

Collection of high-performance multi-threaded malloc() implementation and performance analysis tools.

- uses statistical sampling

Rust project that is using gperf tools - **cpuprofiler**
https://github.com/AtheMathmo/cpuprofiler

- Heap profiler (malloc)
- Cpu profiler
- Heap leak-checker

Visualizer tools:

- GraphViz
- pprof

https://github.com/gperftools/gperftools

Other tools

CPU bound problems:

toplev CPU performance (counting) tool

https://github.com/andikleen/pmu-tools/wiki/toplev-manual

Off-CPU analysis:

eBPF an universal in-kernel virtual machine, that has hooks all over the kernel.

http://www.brendangregg.com/blog/2019-01-01/learn-ebpf-tracing.html

http://www.brendangregg.com/offcpuanalysis.html

Monitoring framework:

Vector An on-host performance monitoring framework which exposes hand picked high resolution metrics to every engineer's browser.

https://github.com/Netflix/vector

Discussion

- Don’t try to over optimize the program!
- Kachegrind tip page: When all functions take almost equal time and the program feels fast for you: stop! There is no reason to make the program 1% faster with days of optimization.
- Know what the issue is / narrow down the problem before using any tool!
- Know your program and do some expectations for different parts of the code!
- Do not use random tools!
- Do not tune things at random until problem goes away!
- Do not blame someone else, measure!

https://www.youtube.com/watch?v=FJW8nGV4jxY
https://www.youtube.com/channel/UCxzogoHFu8-1H9u522wVQpQ