Valgrind for DragonFly/Net/ Open BSD?

FOSDEM'26

Paul Floyd, Sat 31 2026

Valgrind Overview

Valgrind Overview

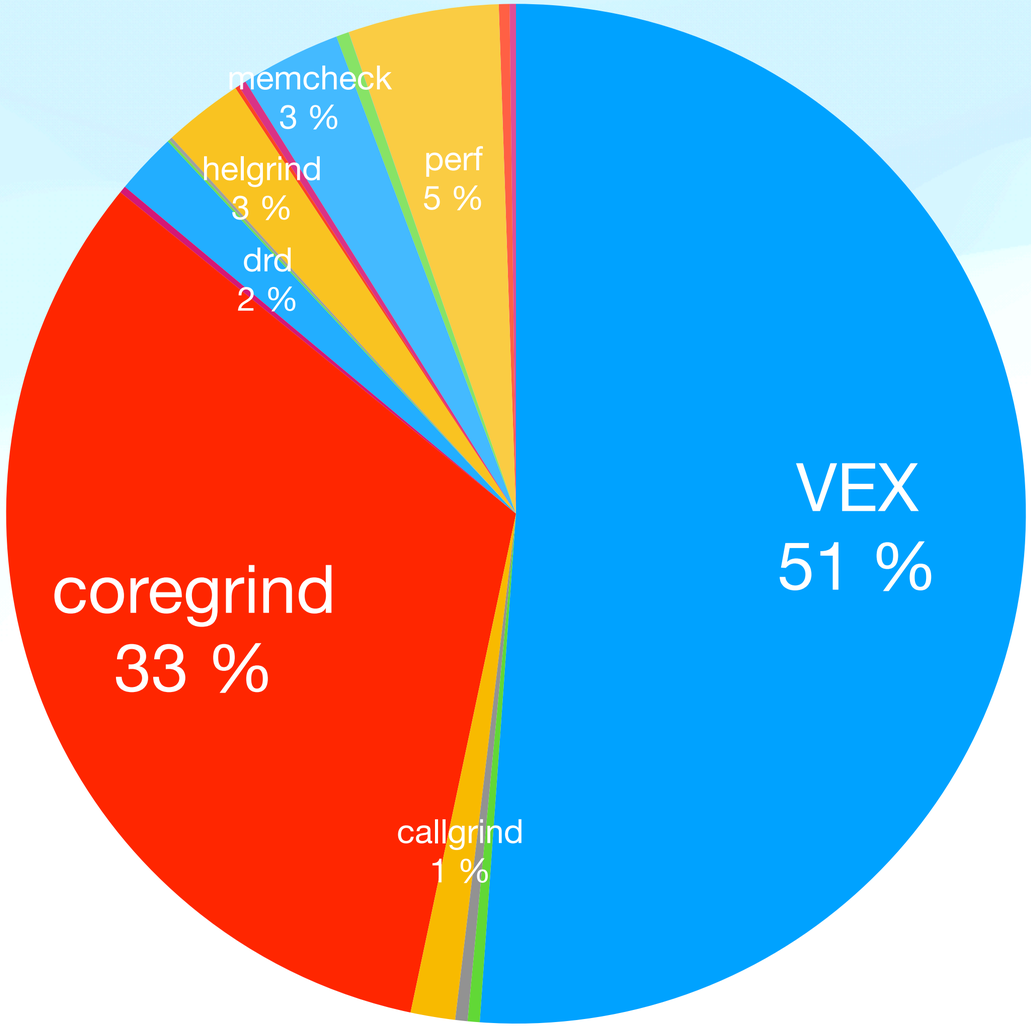
Memory fault detector, profiler and thread hazard detector

- DBA, DBI, dynamic recompilation
- In the process there is the host (the Valgrind tool) running on the physical CPU and the guest or client running on the emulated Valgrind CPU VEX
- Reads guest instructions, converts them to IR, adds instrumentation, does a quick optimisation and regenerates machine code
- Single process, not separate processes like LLDB and GDB
- Tool exe is statically linked - no libc

Valgrind Code

565kloc code [mostly C + some asm] + 275kloc tests

libc replacement
syscalls
memory manager
signals
loader
DWARF
GDB server



amd64
i386
arm
arm64
ppc32
ppc64
mips32
mips64
riscv64
s390

Startup - ELF, DWARF, mmap recording

```
Program Header:
    LOAD off    0x0000000000000000 vaddr 0x0000000000200000 paddr 0x0000000000200000 align 2**12
         filesz 0x0000000000001414 memsz 0x0000000000001414 flags r--
    LOAD off    0x0000000000001420 vaddr 0x0000000000202420 paddr 0x0000000000202420 align 2**12
         filesz 0x0000000000000c70 memsz 0x0000000000000c70 flags r-x
    LOAD off    0x0000000000002090 vaddr 0x0000000000204090 paddr 0x0000000000204090 align 2**12
         filesz 0x00000000000001e0 memsz 0x0000000000000f70 flags rw-
    LOAD off    0x0000000000002270 vaddr 0x0000000000205270 paddr 0x0000000000205270 align 2**12
```

- No ELF = guest exe won’t load
- Bad mmap recording = Valgrind confused about who owns what memory, usually very bad
- No DWARF = no file and line, major usability impact

Hackiness - signals and threads

- Many hacks to get signal handlers to run in VEX and to return to VEX
- Even more hacks to get threads to run in VEX
- Much juggling of signal masks

syscall wrappers - easy

Well, most are easy

```
// SYS_fstatat 552
// int fstatat(int fd, const char *path, struct stat *sb, int flag);
PRE(sys_fstatat)
{
   PRINT("sys_fstatat(%"FMT_REGWORD"d, %#"FMT_REGWORD"x(%s), %#"FMT_REGWORD"x, %"FMT_REGWORD"d )",
         SARG1, ARG2,(char*)ARG2, ARG3, SARG4);
   PRE_REG_READ4(int, "fstatat",
                 int, fd, const char *, path, struct stat *, sb, int, flag);
   ML_(fd_at_check_allowed)(SARG1, (const HChar*)ARG2, "fstatat", tid, status);
   PRE_MEM_RASCIIZ( "fstatat(path)", ARG2 );
   PRE_MEM_WRITE( "fstatat(sb)", ARG3, sizeof(struct vki_stat) );
}
```

- Process related syscalls are difficult

Regtest

- 600 - 800 non-OS-specific related tests
- About 100 OS related tests for Linux and FreeBSD
- No OS related tests for DragonFly/Open/Net BSD

- Also auxchecks (GSL based tests) and LTP (Linux only)
- perf test

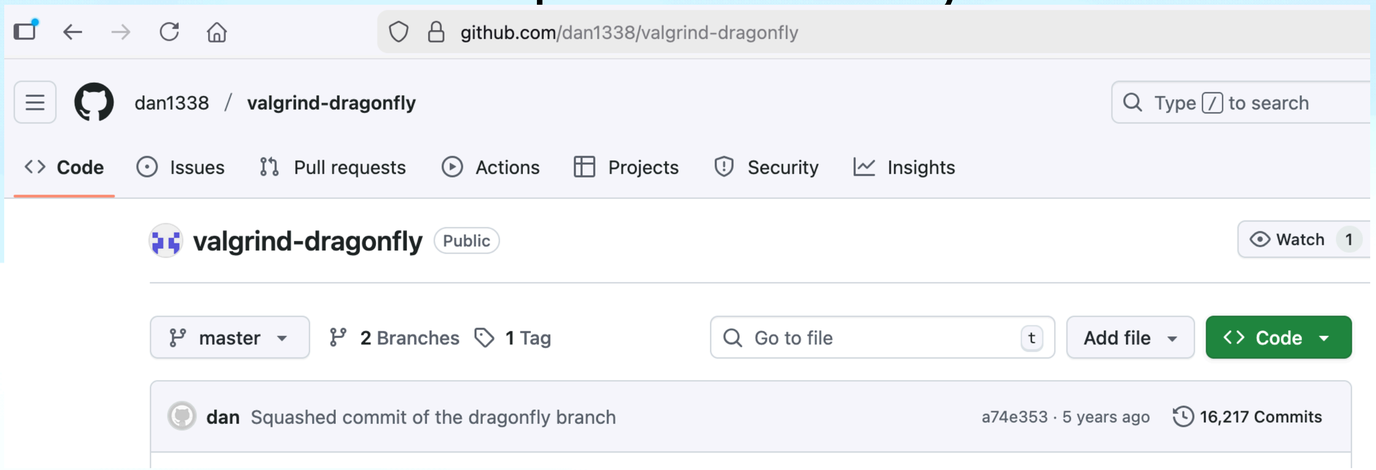
Valgrind developers and infra

- 6 active developers, mostly at RedHat or IBM
- Infra has a lot of history
  - Most hosted by sourceware.org (git, web site, buildbot, forgejo)
  - KDE hosts bugzilla
  - Sourceforge hosts the mailing lists
  - GCC server farm
- Nightly tests triggered every day that there is a git commit

The BSDs

DragonFly BSD

- Work done on the port in 2021 by ‘dan1338’



- Similar to state on FreeBSD at the same time
- Most similar to FreeBSD - should be the easiest

NetBSD

- NetBSD wiki refers to berlios.de via WayBackMachine, no SVN
- pkgsrc only has Linux version
- Project carried out by 4 students at Western Washington University in 2022 https://github.com/WWU-VG4NBSD/valgrind/tree/netbsd-rebase (early dev version of Valgrind 3.20)

PIE = crash

```
netbsd$ ./valgrind-netbsd/vg-in-place yes
==1852== Memcheck, a memory error detector
==1852== Copyright (C) 2002-2024, and GNU GPL'd, by Julian Seward et al.
==1852== Using Valgrind-3.27.0.GIT and LibVEX; rerun with -h for copyright info
==1852== Command: yes
==1852==
==1852== Jump to the invalid address stated on the next line
==1852==    at 0x3027D0: ???
==1852==  Address 0x3027d0 is in a rw- anonymous segment
```

NetBSD - error writing results files

Tools cannot open log or results files

```
---------------------------- coregrind/m_libcfile.c ----------------------------
index 466136b168..72183ec17a 100644
@@ -918,7 +918,7 @@ void VG_(record_startup_wd) ( void )
       szB += 500;
       startup_wd = VG_(realloc)("startup_wd", startup_wd, szB);
       VG_(memset)(startup_wd, 0, szB);
-#   if defined(VGO_linux) || defined(VGO_solaris)
+#   if defined(VGO_linux) || defined(VGO_solaris) || defined(VGO_netbsd)
       res = VG_(do_syscall2)(__NR_getcwd, (UWord)startup_wd, szB-1);
 #   elif defined(VGO_freebsd)
       res = VG_(do_syscall2)(__NR___getcwd, (UWord)startup_wd, szB-1);
```

OpenBSD

OpenBSD manual page server

Manual Page Search Parameters

Search query: pinsyscalls man apropos

2 - System Calls | All Architectures | OpenBSD-current

PINSYSCALLS(2) System Calls Manual PINSYSCALLS(2)

NAME

pinsyscalls — pin system call entry to precise locations in the address space

SYNOPSIS

#include <sys/types.h>

int
pinsyscalls(*void *start*, *size_t len*, *u_int *pintable*, *int npins*);

OpenBSD - do like perl?

- Perl has Sys::Syscall
- Implemented as a “dispatcher” from syscall to libc wrapper
- Two problems
  - Not all syscalls have libc wrappers
  - Chicken and Egg

    Valgrind uses mmap to load the guest exe
    And about 1800 syscalls before libc.so gets mmap’d

OpenBSD

Only possibility?

- Implement a pinsyscalls table

```
struct whats {
        unsigned int offset;
        unsigned int sysno;
} happening[] __attribute__((section(".openbsd.syscalls"))) = {
        { 0x104f4, 4 },
        { 0x10530, 1 },
};
```

https://nullprogram.com/blog/2025/03/06/

Summary

Before and after prep for this talk

	Version	Repo	Package	Regtest before	Regtest after
FreeBSD	3.26	sourceware	OK	99.8 %	
illumos	3.26	sourceware	OK	99.3 %	
DFlyBSD	3.15	GitHub	Almost	65.2 %	78.6 %
NetBSD	3.20	GitHub	Broken	2.8 %	67.1 %
OpenBSD	3.21	GitHub	Broken	21.9 %	41.1 %

Resources

- GitHub repos

  https://github.com/paulfloyd/valgrind-openbsd

  https://github.com/paulfloyd/valgrind-dragonfly

  https://github.com/paulfloyd/valgrind-netbsd
- There is a “fosdem26” branch for each of these repos
- https://valgrind.org/support/mailing_lists.html

  IRC and the mailing list are the most used informal channels

More Resources

- FOSDEM 2022, Valgrind dev room
- https://archive.fosdem.org/2022/schedule/event/valgrind_freebsd/
- FreeBSD Journal October 2024
- https://freebsdfoundation.org/our-work/journal/browser-based-edition/kernel-development/valgrind-on-freebsd/
- Porting Valgrind to NetBSD and OpenBSD - Masao Uebayashi
- https://www.youtube.com/watch?v=PEPo0PJteaA
- https://www.slideshare.net/slideshow/eurobsdcon2014-valgrindpresentation/43399217
- Valgrind web site
- Git repo README_DEVELOPERS, README.freebsd

Thank you for listening

Extras

FreeBSD history

Real start Jan 2020
First Call For Testing mid Feb 2020
Valgrind commit bit Nov 2020
FreeBSD port updated to Valgrind 3.17 April 2021
FreeBSD support landed upstream Oct 2021
FreeBSD support for arm64 April 2024

Debuggability

- Many tracing and debug and verbose logging options
  - C++ “Hello World” with full logging about 16Mbytes
- Debug with GDB and LLDB works well
- Bonus points if vgdb works
- Extra bonus points for getting Valgrind self-hosting to work

Memory Map

```
Legend: (1,67,3) /usr/bin/yes
0: RSVN 0000000000-00000fffff 1048576 ----- SmFixed
1: file 0000100000-0000100fff    4096 r-x-- F   (1,67)
2:      0000101000-00002fffff 2093056
3: file 0000300000-0000301fff    8192 rw--- F   (1,67)
4: anon 0000302000-0000b01fff 8388608 rw---
```

OpenBSD - no DRD or Helgrind readers

```
---------------------------- include/pub_tool_redir.h ----------------------------
index d47e65295..4d879dd39 100644
@@ -286,7 +286,7 @@
 #elif defined(VGO_freebsd)
 #  define  VG_Z_LIBPTHREAD_SONAME  libthrZdsoZa          // libthr.so*
 #elif defined(VGO_openbsd)
-#  define  VG_Z_LIBPTHREAD_SONAME  libpthreadZdsoZda     // libpthread.so.*
+#  define  VG_Z_LIBPTHREAD_SONAME  libpthreadZdsoZa      // libpthread.so.*
 #elif defined(VGO_darwin)
 #  define  VG_Z_LIBPTHREAD_SONAME  libSystemZdZaZddylib  // libSystem.*.dylib
 #elif defined(VGO_solaris)
```

Scheduler / VEX

Complicated but needs little maintenance