Porting Valgrind on Solaris

Ivo Raisr Petr Pavlů

February 2, 2014

Porting Valgrind on Solaris

- Project started as a diploma thesis
- Gained a traction afterwards
- Currently still work-in-progress with some limitations
- Available at
  `https://bitbucket.org/setupji/valgrind-solaris`
- Maintained as a separate fork from upstream, sync'ed from time to time

Authors

- Collaborative effort of:
- Petr Pavlu – initial author, core functionality (threads, signals, syscall machinery), many tests
- Ivo Raisr – coredump support, syscall, ioctl and door wrappers, vgdb-invoker implementation
- Theo Schlossnagle – initial AMD64 support

Solaris and illumos

- SunOS = Unix operating system developed by Sun Microsystems in 1980's
- Aimed at *SPARC* workstations and server computer systems (centralized computing)
- Rebranded to Solaris in 1992

Solaris and illumos

- Popular stateless thin client *SunRay*
- Popular sparc machines in past:
  - UltraSparc III from Sun Microsystems
  - T1 (Niagara) with hardware threads and cryptographics acceleration
  - M5000 and M9000 from Fujitsu for enterprise workloads

Solaris and illumos

- Solaris 10 released on both sparc and x86 in 2005
- OpenSolaris initiative (”almost” open source with CDDL license) in 2005
- First release of OpenSolaris in 2008
- Several experimental ports of OpenSolaris (apart from x86 and sparc)
- In 2008 Sun Microsystems had problems $\rightarrow$ talks with IBM and HP about merger
- In 2010 Sun Microsystems was acquired by Oracle

Solaris and illumos

- illumos forked right before Oracle announced OpenSolaris decomission (illumos is just OS/Net consolidation)
- illumos used in a number of distributions; most known OpenIndiana
- Oracle Solaris 11 released in 11/2011 after 6 years of development; closed source again

ORACLE

SOLARIS

Solaris and illumos

- Oracle Solaris 11.1 released in 10/2012
- Oracle Solaris 11.2 will be released around mid-2014; supporting new generation of sparc T5 and Fujitsu M6
- Solaris and illumos are no longer desktop OS rather they are designed for highly-scalable engineered systems or appliances
- valgrind-solaris aims at both Oracle Solaris and illumos

Current status

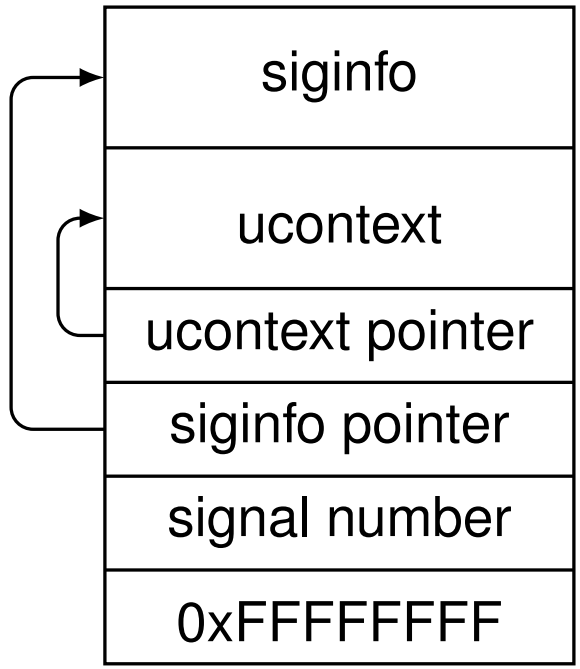
- Sync'ed with upstream valgrind post 3.9.0
- Support for x86 platform stable
- Support for AMD64 in progress
- Tools helgrind and drd are currently disabled
- Test suite results on Oracle Solaris 11.1:
  438 tests, 8 stderr failures, 2 stdout failures, 0 stderrB failures, 2 stdoutB failures, 0 post failures
- A few more failures are present on illumos

Differences between Solaris and Linux

- GNU toolchain used (gcc 4.5 or higher, autotools, gmake) with exception of Solaris link editor (ld)
- Configure-time checks for available functionality
- System calls are not an exported and stable interface – the standard library (libc) is
- Different syscalls (almost all of them)
- Syscall mechanics different
- Threads creation via lwp_create() not sys_clone()
- . . .

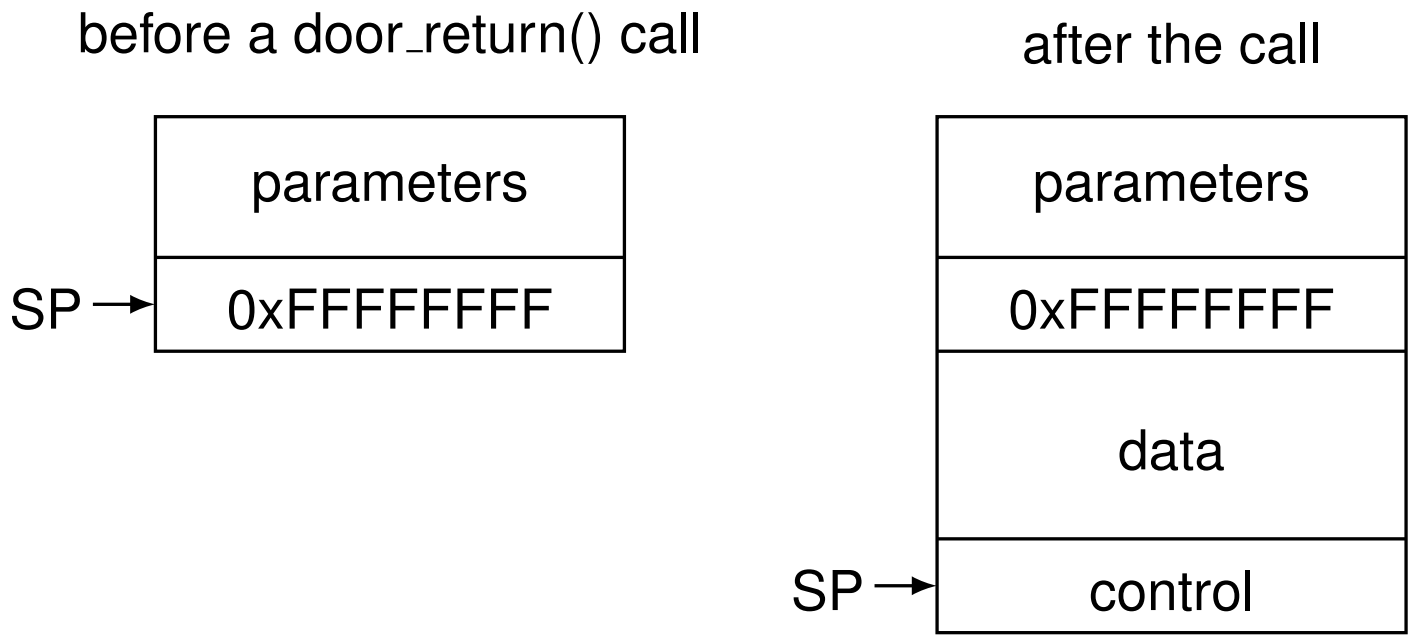
Porting difficulties – Signal handling

- On Solaris, user space is completely responsible for returning from a signal handler (no sa_restorer) and the kernel does not directly restart interrupted syscalls
- Port builds simulated signal frames without Valgrind data (except a few values)



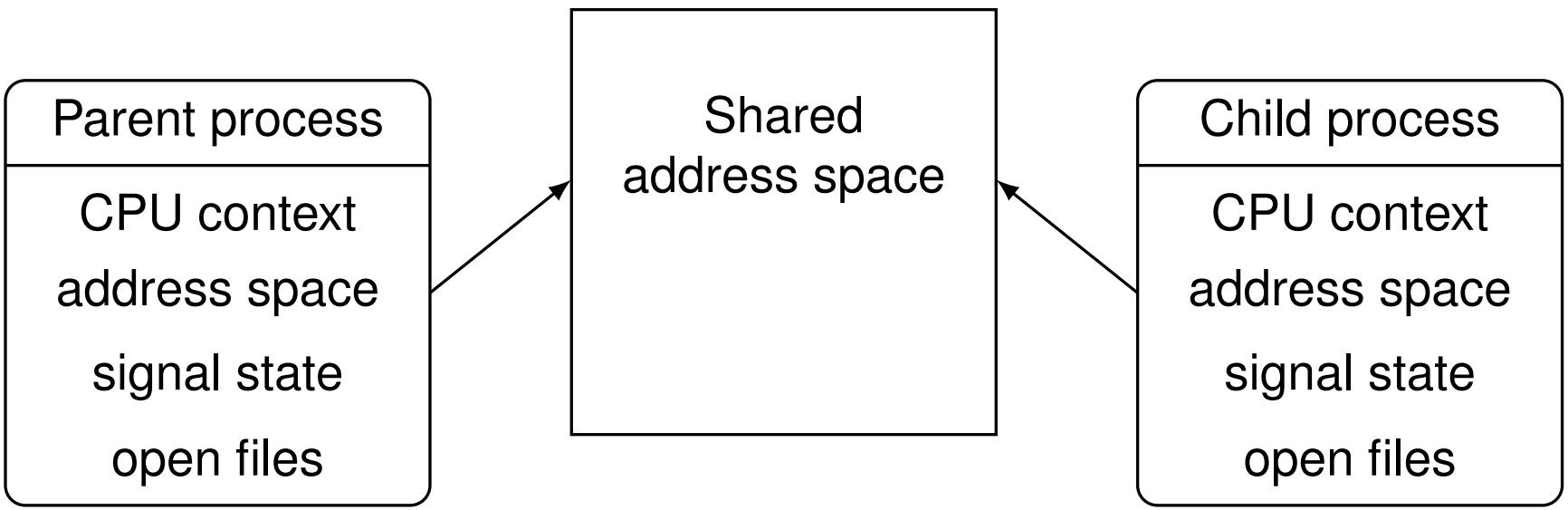
Porting difficulties – Doors facility

- Facility for fast inter-process communication, developed as a part of the Spring operating system in early 1990s
- Server threads receive client data on stack $\rightarrow$ door_return() calls have to be executed while running on the guest stack



Porting difficulties – Vfork support

- On Linux, Valgrind translates vfork to fork
- On Solaris, correct vfork semantics has to be supported, the standard library (libc) relies on it
- Several core parts of Valgrind have to be aware of this support



Sustainability of the project

- When possible, the port tries to have stricter implementation than the upstream code (Linux, Mac OS X); any error caused by a new Solaris version or by changes in the common code should be visible early
- Collaborative effort of several people; each of them can make non-trivial changes in the port
- All port-specific code is thoroughly tested – currently 40 tests with 5000 lines of code in total, over 300 atomic scalar tests