FreeBSD/RISC-V

Ruslan Bukin

University of Cambridge Computer Laboratory

January 5, 2016

Approved for public release; distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

What is FreeBSD?

- Advanced general-purpose Operating System
- Open Source Permissively Licensed Operating System
- UNIX (like), POSIX
- Full system
- Over 30 years of history
  - FreeBSD started in 1993, history to 1972

Who uses FreeBSD ?

- WhatsApp
  - Serves 1 billion of users
  - 1 million active TCP connections per server
- Netflix
  - Serves 1/3 of North America internet traffic
  - Streaming 80 Gbps (90 Gbps on experimental hardware)
- Yahoo!, New York Internet, ISC
- Sony Playstation 4 (Orbis OS)
- Apple (Mac OS)
- Verisign (netmap)
  - Serves root DNS services
- CHERI/Capsicum
- McAfee Research (now Intel Security)
- EMC/Isilon (OneOS)
- NetApp
- Juniper Networks (JunOS)
- .. lots more

FreeBSD architectures

- AMD64
- ARMv7 (Cortex A5-15)
  - Altera, Freescale, Samsung, etc
- ARMv8 (Cortex A53/72)
  - AMD Opteron, Cavium Thunder-X
- MIPS
- PowerPC
- RISC-V (RV64I) – *FreeBSD 11.0*
  - UCB Spike simulator

Some reasons to use FreeBSD

- Full stack BSD license (RISC-V, FreeBSD, LLVM/Clang)
- LLVM/Clang/LLDB
- Technology transition
- Full integrated build system
- Research tools
  - DTrace
  - Perfomance Monitoring Counters
  - Netmap
  - FPGA/ARM heterogeneous SoC tools
- FDT
- UFS2, ZFS
- U-Boot loader (ubldr), UEFI
- Strong code style requirement (style(9))

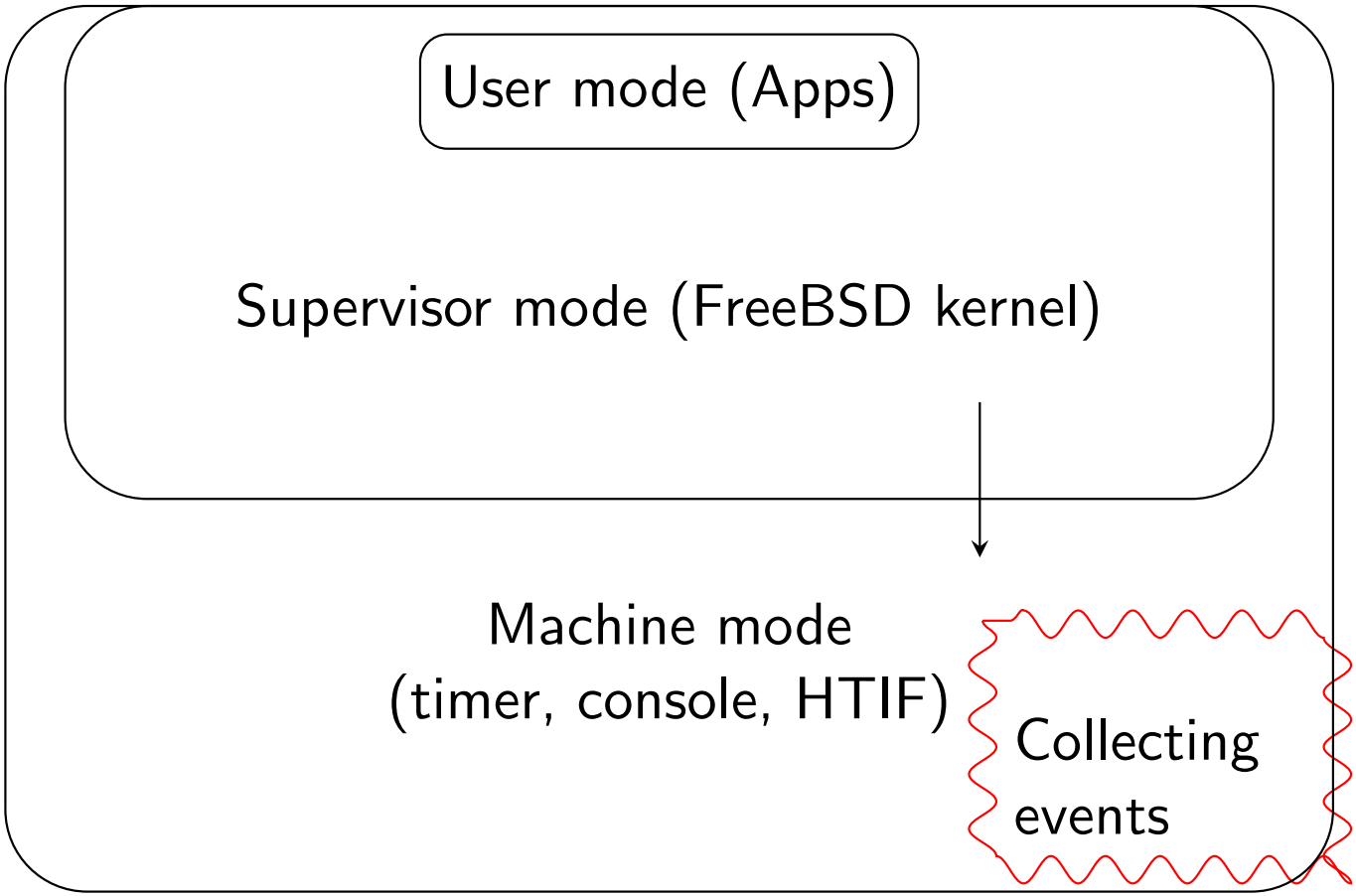
Porting: tools required

- objdump
- elfdump (for runtime linker)
- hexdump (for HTIF block device)

Table: Simulator/Emulator comparison

	Free	Fast	Simple	Advanced
QEMU	No	Yes	Yes	No
Gem5	Yes	No	No	Yes
UCB Spike	Yes	Yes	Yes	No

Machine mode



Early assembly code

1. Puts the hardware into a known state
2. Build a ring buffer for machine mode interrupts
3. Builds the initial pagetables
4. Enables the MMU
5. Branches to a virtual address (exiting from machine mode)
6. Calls into C code

Porting: kernel 1/2

- Early HTIF console device
  - EARLY_PRINTF
- Atomic inline functions
  - atomic_add(..)
  - atomic_cmpset(..)
  - atomic_readandclear(..)
  - ...
- Providing physical memory regions for VM subsystem
  - add_physmap_entry(0, 0x8000000, ..); /* 128 MB at 0x0 */
- VM (pmap)

Porting: kernel 2/2

- Exceptions, context-switching, fork_trampoline
- Timer, interrupt controller drivers
- HTIF block device driver (or use memory disk)
- copy data from/to userspace
  - fubyte, subyte, fuword, suword, sueword, fueword
  - copyin, copyout
- Trying to run /bin/sh (staticaly linked)
- Signals

FreeBSD/RISC-V: Exceptions (1/2)

```
supervisor_trap:
  csrrw sp, mscratch, sp
  ..
  csrr     t0, mcause
  bltz     t0, machine_interrupt
  ..
  la t2, cpu_exception_supervisor
  csrw   stvec, t2
  ..
  csrrw sp, mscratch, sp

 /* Redirect to supervisor */
 mrts

machine_interrupt:
  ..
```

Machine trap vector

```
cpu_exception_
      supervisor()
```

Supervisor mode

FreeBSD/RISC-V: Exceptions (2/2)

```
ENTRY(cpu_exception_supervisor)
        save_registers 1
        mv      a0, sp
        call    _C_LABEL(do_trap_supervisor)
        load_registers 1
        eret
END(cpu_exception_supervisor)

ENTRY(cpu_exception_user)
        csrrw   sp, sscratch, sp
        save_registers 0
        mv      a0, sp
        call    _C_LABEL(do_trap_user)
        load_registers 0
        csrrw   sp, sscratch, sp
        eret
END(cpu_exception_user)
```

FreeBSD/RISC-V: VM (pmap)

- Most sensitive machine-dependent part of VM subsystem
- Around 40 machine-dependent functions for managing page tables, address maps, TLBs
  - pmap_enter(pmap_t pmap, vm_offset_t va, vm_page_t m, vm_prot_t prot, u_int flags, int8_t psind)
  - pmap_extract(..)
  - pmap_remove(..)
  - pmap_invalidate_range(..)
  - pmap_remove_write(..)
  - pmap_protect(..)
  - pmap_activate(..)
  - pmap_release(..)
  - pmap_unwire(..)
  - ...

FreeBSD/RISC-V: Context switching

```
/* a0 = old thread, a1 = new thread */

ENTRY(cpu_switch)
la      x14, pcpup                      /* Load PCPU */
sd      a1, PC_CURTHREAD(x14)           /* Replace thread*/
/* Save old registers */
ld      x13, TD_PCB(a0)                 /* Load old PCB */
sd      sp, (PCB_SP)(x13)               /* Store sp */
sd      ra, s[0-11], ...
/* Switch pmap */
ld      x13, TD_PCB(a1)                 /* Load new PCB */
ld      t0, PCB_L1ADDR(x13)
csrw    sptbr0, t0
/* Load new registers */
ld      sp, (PCB_SP)(x13)               /* Load sp */
ld      ra, s[0-11], ...
END(cpu_switch)
```

Porting: userspace

- jemalloc
- csu
  - crt1.S, crtn.S, crti.S
- libc
  - syscalls
  - setjmp, longjmp
  - _set_tp
- msun
- rtld-elf (runtime-linker)

Porting: syscalls (1/2)

Userspace

```
ENTRY(__sys_mmap)
        li      t0, 477
        ecall
        bnez    t0, cerror
        ret
END(__sys_mmap)
```

Kernel

```
switch(exception) {
        ..
case EXCP_USER_ENV_CALL:
        syscallenter(frame);
        break;
};
        ..
```

Porting: syscalls (2/2)

```
int
cpu_fetch_syscall_args(frame, ..)
{
        ..
        ap = &td->td_frame->tf_a[0];
        ..
}

void
cpu_set_syscall_retval(frame, int error)
{
        ..
        frame->tf_t[0] = error;
        ..
}
```

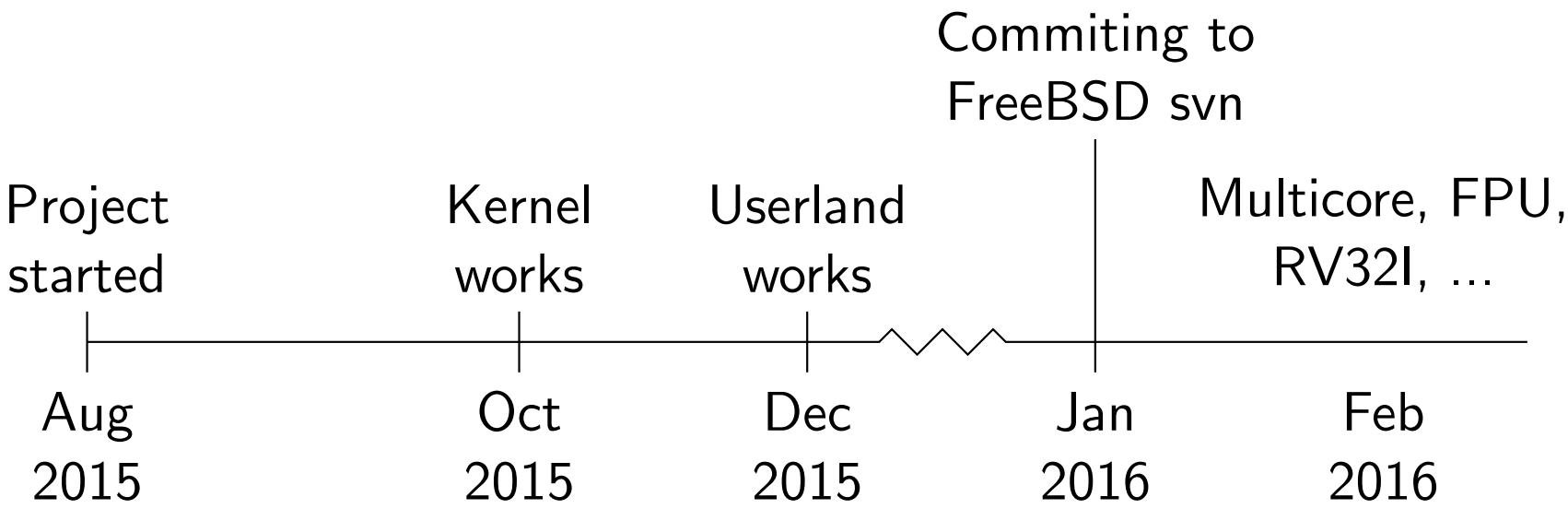
FreeBSD/RISC-V: facts

- based on ARMv8 port
- diff 25k lines (200 new files)
- 6 months from scratch

People involved (Thanks!)

- Robert Watson (University of Cambridge)
- David Chisnall (University of Cambridge)
- Andrew Turner (ABT Systems)
- Arun Thomas (BAE Systems)
- Ed Maste (The FreeBSD Foundation)

FreeBSD/RISC-V: Current status



FreeBSD/RISC-V: Current port status

- csu (**C** **s**tart **u**p) – *committed*
- machine headers – *committed*
- rtld-elf (runtime linker) – *committed*
- libc
- libthread
- kernel
- userspace (rest of)

} January 2016

FreeBSD/RISC-V: Next plans

- Multicore
- Floating Point Unit (FPU)
- TLB cache
- HTIF ethernet device
- Increase VA space
- Hardware bringup (including FPGA implementations)
- RV32I
- DTrace
- Perfomance Monitoring Counters ?
- QEMU
- Separate machine mode code
- FreeBSD ports/packages

Change proposed: split SPTBR

Split sptbr to sptbr0 and sptbr1 for user VA and kernel VA respectively, that gives

- No need to change SPTBR any time changing privilege level
- Reduce code size
- Avoid mess in the code

split SPTBR: example

Example: Spike simulator

Before:

```
-   reg_t base = proc->get_state()->sptbr;
```

After:

```
+   reg_t base;
+   if ((addr >> 63) == 1) /* Kernel space */
+     base = proc->get_state()->sptbr1;
+   else { /* User space */
+     base = proc->get_state()->sptbr0;
+   }
```

Questions

Project home:
https://wiki.freebsd.org/riscv