Introduction to Bhyve

FreeBSD Fridays

Peter Grehan grehan@freebsd.org

What is bhyve ?

- Short for **B**SD **Hy**per**v**isor
- Base-system hypervisor for FreeBSD/amd64
- Uses Intel VT-x/AMD-SVM h/w acceleration
- Run Windows/Linux/*BSD virtual machines at close-to native speed

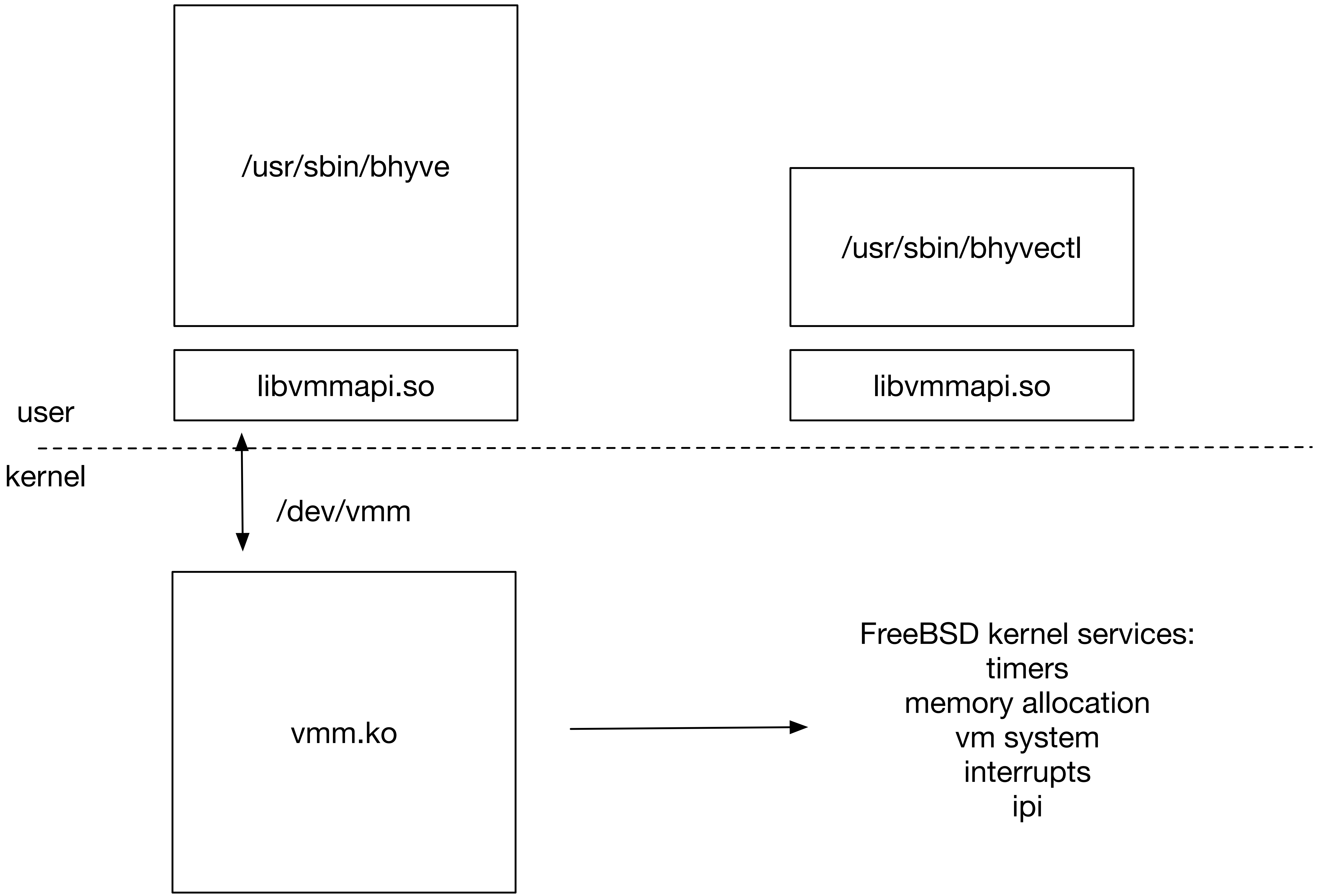
What this talk will cover:

- Origins
- bhyve components
- How CPU/Memory/IO virtualization works with bhyve
- Monitoring
- Resources

Short history

- Developed as a NetApp internal project, late 2010
- Coincided with Intel's “EPT” memory virtualization addition to VT-x
  - Hypervisor implementation effort massively reduced
- Code donated to FreeBSD - May 2011
- AMD SVM support added - Oct 2014
- Windows guest support - May 2016

bhyve components



/usr/sbin/bhyve

/usr/sbin/bhyvectl

libvmmapi.so

libvmmapi.so

user

kernel

/dev/vmm

vmm.ko

FreeBSD kernel services:
timers
memory allocation
vm system
interrupts
ipi

Virtualization: CPU

- A virtual machine is represented with a `/usr/sbin/bhyve` process
- 1 thread per virtual CPU (vCPU) in this process
    - vCPU initialized to power-on state
- ioctl() to the `vmm.ko` kernel module for context switch to guest state
- SMP system represented by multiple vCPU threads

Virtualization: Memory

- Guest operating systems given the illusion of contiguous physical memory
- But, guest physical memory translated to host physical memory using “second level” page tables
    - Allows non-contiguous host memory to be used
    - Page faults provide on-demand allocation of guest memory
- FreeBSD virtual memory system modified to support second-level page tables.
- Guest memory backed by swap space
    - can even be swapped out when the system under memory pressure

Virtualization: Input/Output

- Intel/AMD h/w assist provides CPU and memory virtualization
- I/O virtualization done entirely* in software by the hypervisor
- Most of `/usr/sbin/bhyve` dedicated to this
- Guest operating systems given the illusion of running on an x86 PC
    - timers, interrupts, serial ports, PCIe busses and adapters

* there’s always an exception

I/O: timers

- bhyve emulates a number of timers and clock sources
    - 8254 PIT
    - HPET
    - local APIC timer
    - ACPI PM timer
    - IBM PC-compatible RTC
- These are implemented in `vmm.ko` to reduce jitter and to use kernel high-resolution timers.

I/O: PCI

- A PCI hierarchy is emulated
- Virtual adapters are placed at slots in this hierarchy
- Legacy, MSI and MSI-x interrupts are supported for adapters
- Large number of adapters can be configured
    - Theoretical max: 256 buses, 32 slots/bus, 8 functions/slot

I/O: Network

- Emulations available for:
  - Intel 82545 Gb Ethernet (e1000)
  - Virtio network device (virtio-net, higher performance)
- Adapter MAC address auto-generated (manual override allowed)
- Ethernet frames sent/to from a `tap(4)` device on the host
  - `bridge(4)` can be used to connect `tap` devices to the outside world

I/O: Disk

- Emulations available for
  - AHCI (disk and CDROM)
  - NVMe
  - Virtio block device
  - Virtio SCSI
- Backing disk image can be a file or a block device
- 8 threads per adapter to improve concurrency

I/O: Graphics console

- bhyve provides a simple frame buffer for guest graphical output
  - Up to 1920 x 1200 x 32bpp
- The UEFI firmware has support for this device
- Input via PS2 keyboard/mouse emulation
  - Also, USB tablet
- No base-system graphics in FreeBSD, so…
  - tiny VNC server provided in bhyve

I/O: PCI passthru

- Exception to “all done in software” for device emulation
- Allows a hardware adapter to be assigned directly to a guest
    - With a constraint: the adapter must support MSI/MSI-x interrupts.
- The host I/O memory-management unit (IOMMU) is used to direct adapter DMA to the correct memory locations on the host.
- Requires guest memory to be pre-allocated (“wired”)
- Useful when
    - highest performance required
    - an adapter isn’t supported by FreeBSD

I/O: miscellaneous

- Sound
    - Intel HDA audio device emulated
    - Uses `/dev/dsp` on the host for output and mic
- Serial
    - Up to 4 16550A devices emulated
    - Any host tty device can be used (including stdio)
- Random
    - Virtio random device
    - Backed by `/dev/random`

Firmware

- Two ways to boot a guest:
  1. UEFI firmware image
     - Custom build of open-source EDK2, with custom bhyve drivers
     - Supports graphical boot
     - “CSM” variant to support BIOS boot
  2. Direct kernel boot
     - `/usr/sbin/bhyveload`, or `grub-bhyve` from ports
     - Serial console only

Putting it all together

- to install a Windows 10 guest of a CDROM image, 2 vCPUs, 4GB RAM

```
bhyve \
 -c 2 \
 -s 0,hostbridge \
 -s 3,ahci-cd,/images/en_win10_distro.iso \
 -s 4,e1000,tap0 \
 -s 11,fbuf,tcp=0.0.0.0:5900,wait \
 -s 20,xhci,tablet \
 -s 21,nvme,/images/win10.img \
 -s 31,lpc \
 -l bootrom,/usr/local/share/uefi-firmware/BHYVE_UEFI.fd \
 -m 4G -H -w \
w10
```

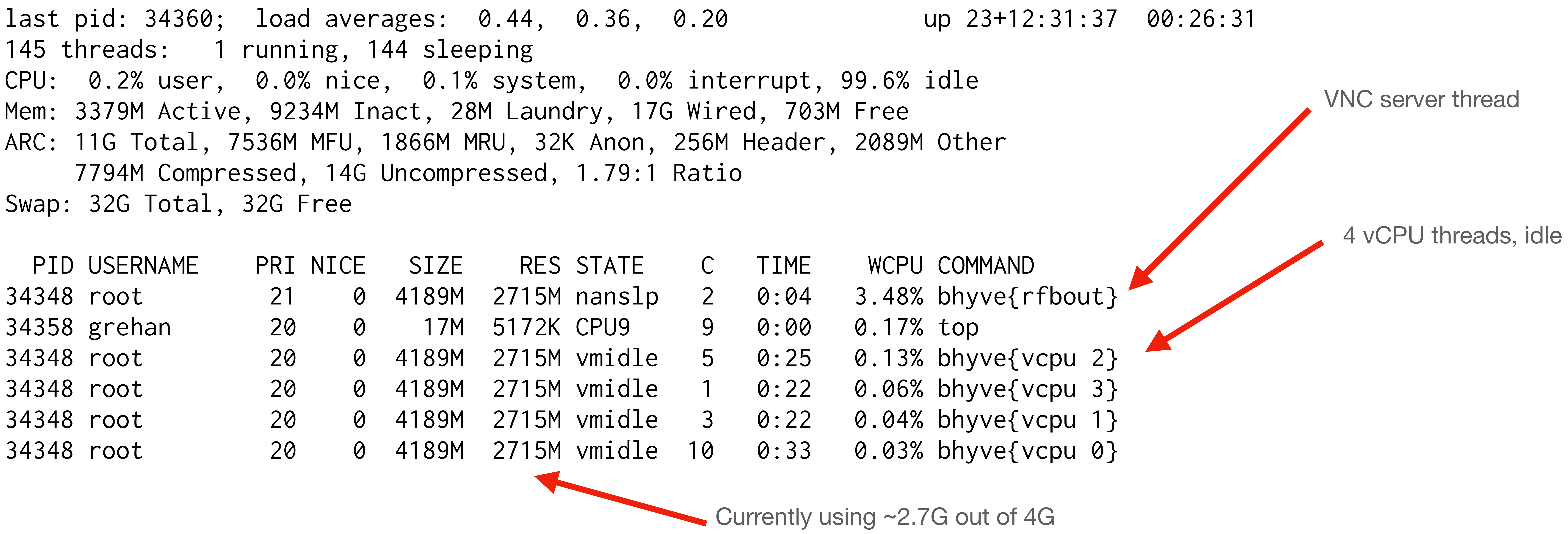
Simplifying using a front-end

- A number of tools exist to ease configuration
- vm-bhyve from the ports collection is popular
- Windows 10 installation:

```
vm create -t windows w10
vm install winguest en_win10_distro.iso
vm start w10
```

Monitoring

- bhyve can be monitored using existing FreeBSD utilities
  - ps, top, gstat, tcpdump, dtrace
- “top -H” recommended; vCPUs and i/o threads shown, as well as memory.



```
last pid: 34360;  load averages:  0.44,  0.36,  0.20             up 23+12:31:37  00:26:31
145 threads:   1 running, 144 sleeping
CPU:  0.2% user,  0.0% nice,  0.1% system,  0.0% interrupt, 99.6% idle
Mem: 3379M Active, 9234M Inact, 28M Laundry, 17G Wired, 703M Free
ARC: 11G Total, 7536M MFU, 1866M MRU, 32K Anon, 256M Header, 2089M Other
     7794M Compressed, 14G Uncompressed, 1.79:1 Ratio
Swap: 32G Total, 32G Free

  PID USERNAME    PRI NICE   SIZE    RES STATE    C   TIME    WCPU COMMAND
34348 root         21    0  4189M  2715M nanslp   2   0:04   3.48% bhyve{rfbout}
34358 grehan       20    0    17M  5172K CPU9     9   0:00   0.17% top
34348 root         20    0  4189M  2715M vmidle   5   0:25   0.13% bhyve{vcpu 2}
34348 root         20    0  4189M  2715M vmidle   1   0:22   0.06% bhyve{vcpu 3}
34348 root         20    0  4189M  2715M vmidle   3   0:22   0.04% bhyve{vcpu 1}
34348 root         20    0  4189M  2715M vmidle  10   0:33   0.03% bhyve{vcpu 0}
```

VNC server thread

4 vCPU threads, idle

Currently using ~2.7G out of 4G

Futures

- ARM64 port
- Virtio 9p filesystem access
- Save/resume
- GPU passthru

Project Resources

- FreeBSD Handbook
  https://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/virtualization-host-bhyve.html

- FreeBSD wiki
  https://wiki.freebsd.org/bhyve

- freebsd-virtualization email list
  https://lists.freebsd.org/mailman/listinfo/freebsd-virtualization

- Bhyve office hours
  Will be announced on http://live.freebsd.org

Thank you !