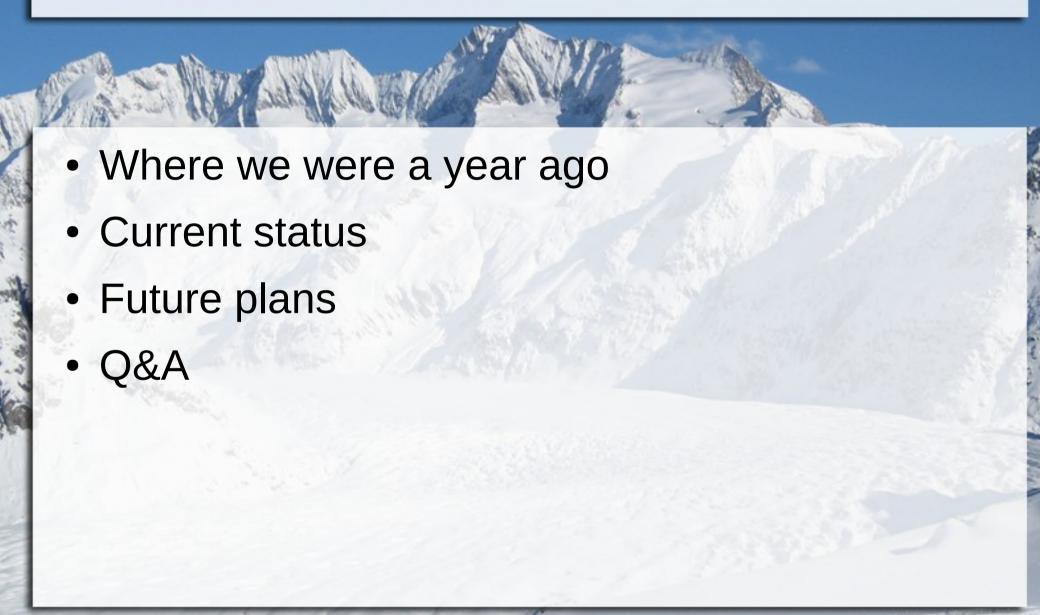
OpenBSD vmm/vmd Update



Agenda



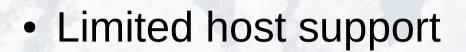
- One year ago, I demoed vmm(4) + vmd(8) here
 - Basic VMX operation, limited features

- Lack of important platform devices
- Poor interrupt handling
- Limited VM control
- Support for only minimal host variety

- Lack of important platform devices
 - No PIC (interrupt controller)
 - Hardwired interrupt priority
 - Dropped lower priority interrupts
 - No PIT (hardware timer)
 - vmm(4) injected a hardclock every 8000 VM entries
 - Only minimal vio(4), vioblk(4) and com(4) support



- vmctl(8) supported only basic start/stop/status operations
- No reboot
- No graceful shutdown
- No support for vm.conf(5) configuration file
 - (well, sort-of)



- amd64 hosts only
- Intel only
- Unrestricted guest capability required
 - Provides ability to virtualize real mode
 - Requires Westmere or later CPU

This Past Year ...

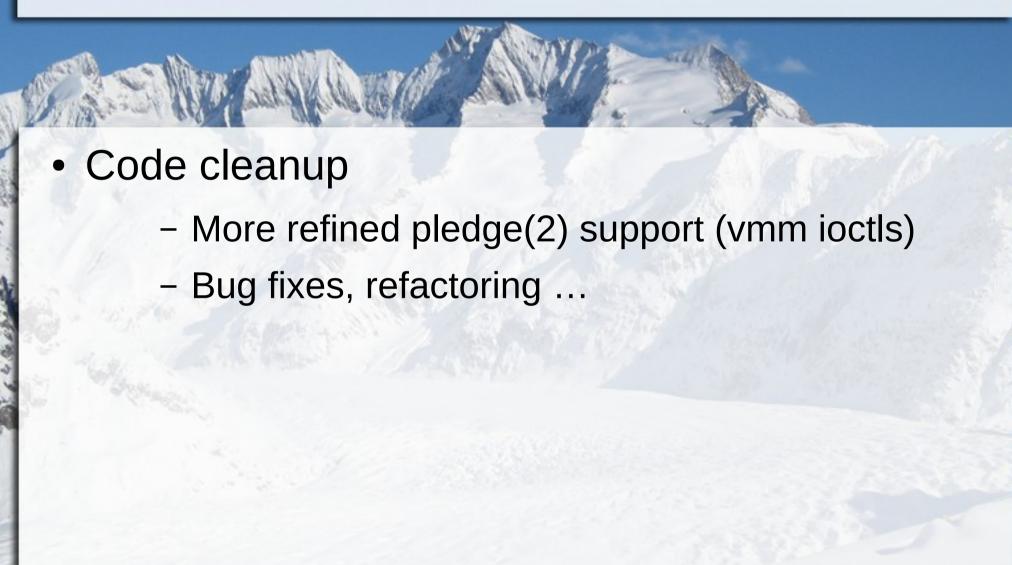
- What we lacked in features last year, we made up for in enthusiasm and interest
 - A few new (and old) developers became vmm and vmd hackers
 - Support and encouragement from the community has been great
- We've improved quite a few things ...

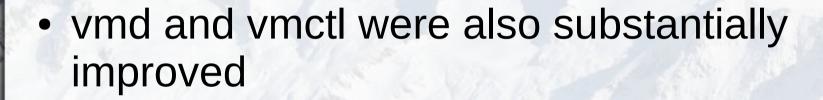
- Solidifying the device model
 - Proper interrupt control
 - PIT timer implementation
 - RTC clock implementation
 - Previously this was passthrough

 Most of this work occurred at the Nantes hackathon in April 2016

- Better platform support
 - Resurrected and merged old i386 vmm tree
 - (To support i386 hosts)
 - Resurrected and merged old SVM tree
 - (To support AMD hosts)
 - Added support for i386 guest VMs
 - Removed requirement for unrestricted guest mode

- Better platform support
 - vmm/vmd shared memory map
 - Required improvement in uvm layer
 - Removed bounce buffers for vio(4) and vioblk(4) queue processing via vmd
 - Support VMs with > 2GB RAM
 - Limit is now MAXDSIZ (32GB currently on amd64)





- Better security
 - More consistent and thorough use of pledge(2)
 - More privsep
 - Use fork+exec model

In particular, VM configuration is far easier now

 VMs do not need to be configured on the command line anymore:

```
# vmctl start OpenBSD_i386_VM -c -k bsd.i386 \
    -m 1024M -i 1 -d /var/vmm/i386_1.img \
    -d /var/vmm/i386_2.img
```

In particular, VM configuration is far easier now

 VMs do not need to be configured on the command line anymore:

```
# vmctl start OpenBSD_i386_VM /c -k bsd.i386 \
    -m 1024M -i 1 -d /var/vmm/i386_1.img \
    -d /var/vmm/i386_2.img
```

In particular, VM configuration is far easier now

VM creation can be streamlined:

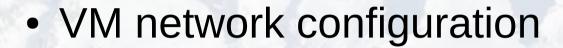
```
# vmctl start OpenBSD_i386_VM
```

• In particular, VM configuration is far easier now

- VM creation can be streamlined
- This works for non-root users as well now:

```
[/home/mlarkin] $ vmctl start OpenBSD_i386_VM
```

- vm.conf(5) can define instantiation rules for VMs:
 - All previous vmctl(8) command line parameters
 - "-k" / "kernel" option no longer needed
 - Permitted control users/groups
 - Network configuration



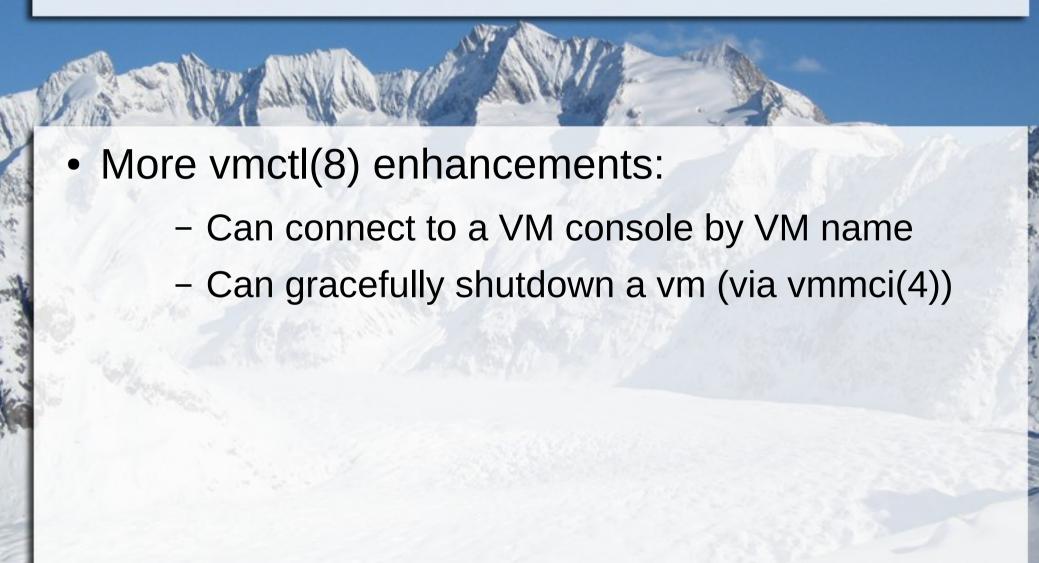
- vmd can now autoconfigure switch(4) and bridge(4) interfaces when starting VMs
- On VM start, vmd can create a new switch for a group of VMs, or automatically add specific VM interfaces to a switch/bridge
- Interface groups can also be optionally created (for easy integration into pf.conf, for example)

 Host-side interfaces are also tagged with the owning VM in ifconfig(8) output

- vm.conf can also contain rules governing who can manipulate a VM
 - ... by username
 - ... by group
- Allows owners or members of owning groups start/stop controls for that VM
 - ... except if root launches the VM first
- VMs can also be autostarted at system boot or defined but not started automatically

 vmctl(8) shows status of all VMs defined in vm.conf, with their owners:

```
# vmctl status
ID PID VCPUS MAXMEM CURMEM TTY OWNER NAME
26 28094 1 1.0G 26.5M ttyp4 :mlarkin firefox
25 37862 1 1.0G 169M ttyp1 root amd64
- - 1 1.0G - :mlarkin bhyvecon2017
```



vmmci(4)



- Provides host → vm communication
- Modeled after vmt(4) but implemented as an OpenBSD specific PCI virtio device
- Handles shutdown and reboot requests (graceful shutdown)

In Process



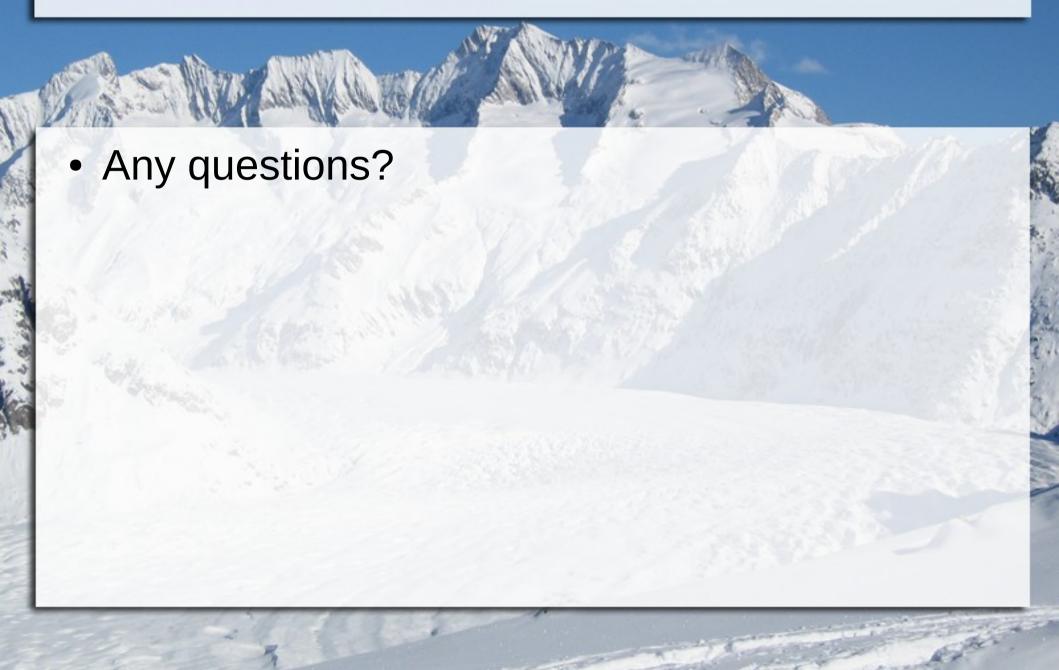
- Finish merging SVM tree
 - Implement interrupt windowing support
- Pay off some "bug debt"
- Start merging the two remaining old trees
 - Shadow paging (very out of date now)
 - Nested VMX (not quite as out of date)
- vmd instruction decoder and memory parser
 - Needed for some non-OpenBSD guests

In Process



- Provide a BIOS
 - Useful for other guest OSes but not strictly required
 - We aim to keep this as an optional add-on
- More platform devices
 - SMP support
- Better non-OpenBSD guest support

Questions?



Thank You

