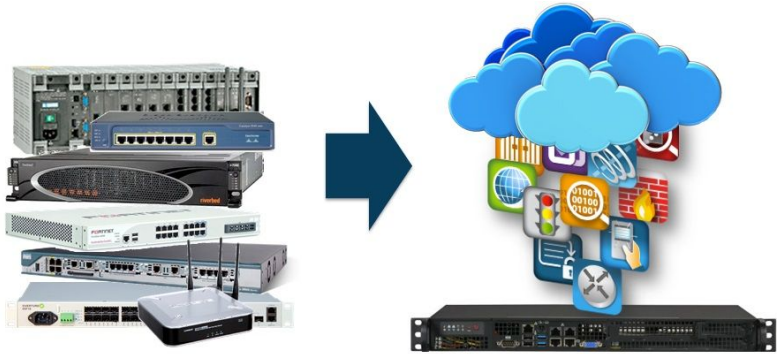
µVirt: Virtualization on OpenWrt

Mathew McBride
<matt@traverse.com.au>
@mcbridematt

Why virtualize?

- “Universal CPE” concept
  - Telco point of view: Standardized (“whitebox”) CPE, (Truck)roll once, deploy many
  - Often as a method of extending private cloud to customer “edge”



https://www.sdxcentral.com/articles/contributed/understanding-use-universal-cpe/2017/07/
Image from article (ADVA Optical Networking / SDxcentral)

Goals

- Demonstrator for small virtualization on ARM64
- Particular emphasis on “Universal CPE” use case
  - Customer sites with “appliance” spec boxes (typical 4-16GB RAM, <=256GB SSD)
  - Typical setup: Firewall, VoIP, IDS/IPS, SD-WAN VM’s
- Easy to use - works standalone
  - vs OpenStack, Industry (MANO) or commercial NFV stacks.
- Playground for end-to-end solutions
  - Working towards a demonstrator involving central management, SD-WAN/VPN, remote IPMI and full life cycle provisioning via LTE
- Would like to make advanced acceleration techniques available while still integrating with existing OpenWrt config structures

Other use cases

- Deploying value add applications to existing fleet
  - E.g Home automation / Smart Home, media servers for residential CPE
  - Some carriers' residential CPE are in the "micro" uCPE class already
- Multi-tenant virtualized router for MDUs
- Home router and server in a box
  - e.g OpenWrt + NextCloud
- Run software too complex for OpenWrt
- Isolation via VMs

Why on OpenWrt?

- Small footprint
  - Fitting inside unmanaged flash (NOR/NAND) provides BOM savings
    - 128MB,1G,64GB,> 128GB price/technology barriers
  - More storage, RAM and compute can be dedicated to providing ‘revenue-generating’ service than scaffolding
- Built in configuration system
- Good set of layer2 network, process, firewall features
- Image based deployment - sysupgrade
  - Possibility to “rollback” to known good state
  - Avoids issues with apt/rpm breakage in the field
- OpenWrt is our preferred platform :)

Configuration

```
$ cat /etc/config/virt

config vm 'fedora'
        option type 'aarch64'
        option cpu 'host'
        option memory '1024'
        option numprocs '1'
        option telnet '4446'
        option machine 'virt,gic_version=2'
        list mac '52:54:00:A1:51:C0'
        list disks 'fedoradisk'
        list network 'lan'
        option enable '1'
        option provisioned '1'

config disk 'fedoradisk'
        option path '/dev/ssd/fedora'
        option type 'virtio-blk'
```

What runs on uVirt?

- “ARM Server” compliant operating systems (EFI)
  - ARM64 Linux distributions with EFI support
    - Debian/Ubuntu/Fedora/CentOS work out of the box
  - Proof of concept ports (by Traverse):
    - OverTheBox (OpenWrt)
    - Untangle NGFW (Debian)
    - Rockstor NAS (CentOS)
    - Alpine Linux
  - IPFire (in progress)
  - OpenBSD and FreeBSD
    - Catching up to Linux in ARM Server support
- Docker will run on the major distributions

OpenWrt on virtual (or physical) ARM Server

- Submitted ‘armserver’ port earlier - did not go anywhere
  - Optimistic.. Even I don’t have a proper ARM server (ThunderX/Ampere/Centriq) box
  - Some suggestions to add as EFI variant of armvirt port instead
    - I think this is what I will do
    - Similar to the x86-64 EFI patch currently floating around
      - No need for legacy/MBR on ARM

Management

- Proof of concept OpenWISP addon
  - Including support for cloud-init, image download etc.
- Interested in other management systems, i.e TR-069/USP
- Would like to do a LuCI app for uVirt

Current issues

- Suggestions appreciated..
- Lack of visibility of VM ports in LuCI etc.
  - Ideally, should work like WiFi ports as far as network configuration is concerned
- Related: need to fix issues where VM starts up before br-X is ready
  - How to deal with “ifdown lan && ifup lan”?
- Improving VM<->External network performance
  - See DPDK work (next slide)
- Hugetlbfs - used by both VM’s and DPDK
  - Would like to centralize somewhere - /etc/config/system ?

Improving VM forwarding performance

- DPDK + OpenVSwitch
  - Preferred solution from NXP and other vendors
    - Works on our current DPAA1 [LS1043] SoC's, well established on x86
  - Increase in RAM requirements: ~1GB hugetlb
  - Requires own management (OpenFlow)...
- Data Plane/DPAA2 Passthrough (NXP LS1088/2088) (similar to SR-IOV)
  - x86 implementation works more in tandem with external switches
  - DPAA2 is a lot more interesting (L2 switch on dataplane) - Study into this ongoing
  - Would like to build OpenWrt config bindings for this
- Other 'fast path' solutions (see other presentations at OpenWrt Summit)
  - To be evaluated

*We already use vhost-user devices (shortcut bridge->VM)

DPDK Progress

- 17.11 (NXP port) and 18.08 mainline works
  - As far as “testpmd” application
- Proof of concept µVirt with DPDK-OpenVSwitch
  - At the time of writing.. Stuck with segfault issues on the OVS side
- Hope that next DPDK LTS (18.11) and OpenVSwitch will clean up a lot of issues on ARM64
  - NXP tree has a lot of patches to deal with “SoC” (non-PCIe) devices, all have been upstreamed
  - OpenVSwitch policy to work with LTS DPDK only.

Acknowledgements

- External contributions:
  - Thomas Niederprüm - USB pass through, memory ballooning support and improved image builder for Alpine

More information

- https://gitlab.com/traversetech/muvirt
- https://traverse.com.au/application-library/muvirt/

Extra / Backup

Why not <insert here>?

- Built for big end of town
  - OpenStack, OPNFV, MANO etc.
- Heavy system requirements (for systems of this size)
  - Many existing distributions with management systems for KVM
- Too thin or not suitable for self-hosted remote
  - ESXi
    - Wasn’t on ARM64 until recently :)
- Limited hardware support
  - Can use uVirt to run “server” OS’es on smaller boards without EFI or SBAS support

Why DPDK

- QorIQ SoC network complex designed for packet rather than flow acceleration
  - vs. residential gateway accelerators designed to accelerate IPv4 NAT flows etc.
- Significant performance improvements from bypassing Linux NW stack
  - In return for less flexibility
- OpenvSwitch can be accelerated with DPDK
- DPDK improvements work across other comparable ARM SoC's (NXP,Marvell,Cavium), as well as x86

DPDK Challenges and Pitfalls

- Only works on glibc (no musl)
  - We can live with that - system image still <=64MB
  - Caused build issues with other packages, especially missing -lpthread
- Have to give network MACs in their entirety to DPDK
  - tap/tun style interface if you want to access ‘data plane’
  - Hence dedicated ‘management’ ports on router boxes
- CPU scheduling/contention
- hugetlbfs