Vortex from UCloud

NFV in action on enterprise-grade IaaS cloud computing platform

徐亮 leo.xu@ucloud.cn
Agenda

• What and why is UCloud Vortex?
• How to implement it?
• Lessons learned in Ops
What is UCloud Vortex?

- UCloud.cn
  - Top Chinese IaaS service provider
  - 10 worldwide data centers
  - Tens of thousands of enterprise users
- UCloud Vortex
  - A layer-4 load balancing software
  - Just like LVS, but in Cloud scale
    - Multiple tenants
    - Overlay Network
Why?

- Keep It Simple, Stupid!
  - UCloud Vortex = LVS + Multiple Namespaces + OVS
- Easy Customizable
  - Highest Random Weight (HRW) hashing
  - Active - Standby mode backends
- Faster
  - PPS: 14M (10G, 64 bytes line rate)
  - CPS: 200k+
  - Concurrent Connections: 30M+
Performance

The chart shows the performance comparison of LVS, Maglev, and VORTEX across different CPU cores.

- **LVS**: The performance is consistently lower compared to Maglev and VORTEX across all CPU cores.
- **Maglev**: Shows a steady increase in performance with an increase in CPU cores.
- **VORTEX**: Exhibits a significant improvement in performance, especially with more CPU cores.

The x-axis represents the number of CPU cores, ranging from 1 to 7, and the y-axis represents the throughput in Mops (Mega Operations per Second).
How?

• Scale up by share nothing
• RSS
• Multiple-queues TAP device
• Each core owns a session table

• Scale out by ECMP Cluster
How?

• ARP
  • Forward to kernel
  • Get the MAC address of router by Netlink from kernel neighbors table
• Local IP at TAP interface
  • Forward to kernel
  • BGP is handled by Quagga BGPD
• NVGRE
  • Directly forward by Vortex
How?

- HTTPBench
  - A HTTP client and server for Vortex benchmark
  - An Unikernel application with DPDK and Linux TCP/IP stack
  - 50k+ HTTP CPS / core
Libul Architecture

- Inspired by Linux Kernel Library, NS3 DCE, User Mode Linux, RUMP

- Build Linux Kernel to user space static library
- Integrated DPDK with kernel by DPDK netdev
- Export system call handle functions
- Socket Callback VS Epoll
- Application needn’t be changed.
- Application
  - hijack by LD_PRELOAD
  - System Call
  - Socket Callback
  - Socket
    - TCP/IP Stack
    - DPDK netdev
    - ul arch

libhijack.so is optional

Application

Build Linux Kernel to user space static library

Socket Callback VS Epoll

Integrated DPDK with kernel by DPDK netdev

Application needn’t be changed.
Libul Performance

- **OS**: Ubuntu 14.04 (3.13.0-24)
- **CPU**: Xeon E5-2670v2 (10 core 2.5G) * 2
- **NIC**: 82599ES * 2
- **App**: NGINX (Installed by apt-get)
- **Request**: 43 bytes Empty Gif / Connection
Ops - CPU Usage

• CPU usage always 100% in OS
• Calculate CPU load by application
  • Summary effective CPU cycles when received and processes packets
  • CPU load = effective CPU cycles / rte_get_timer_hz
• Adjust by CPU Frequency
Ops - Power Management

- scaling_available_frequencies is not created in /sys
  POWER: File not opened
  POWER: Cannot get available frequencies of lcore 2
  POWER: Unable to set Power Management Environment for lcore 2
- Set “HP Power Regulator” to “OS Control Mode”
- Add “intel_pstate=disable” into kernel command line
Ops - Power Management
Ops - Single Cable Server

- Single Cable Server in legacy IDC
- SRIOV
  - Linux owned PF port
  - DPDK use VF port
- Only 2 queues
  - 4 queues after upgrade Linux ixgbe to 4.1.5
Ops - Troubleshooting

• Dynamic configure dump conditions
  • rte_pktmbuf_dump
    • hard to read
  • forwards to debug tap device
    • lost information, such as input port

• Any Suggestions?
THANK YOU