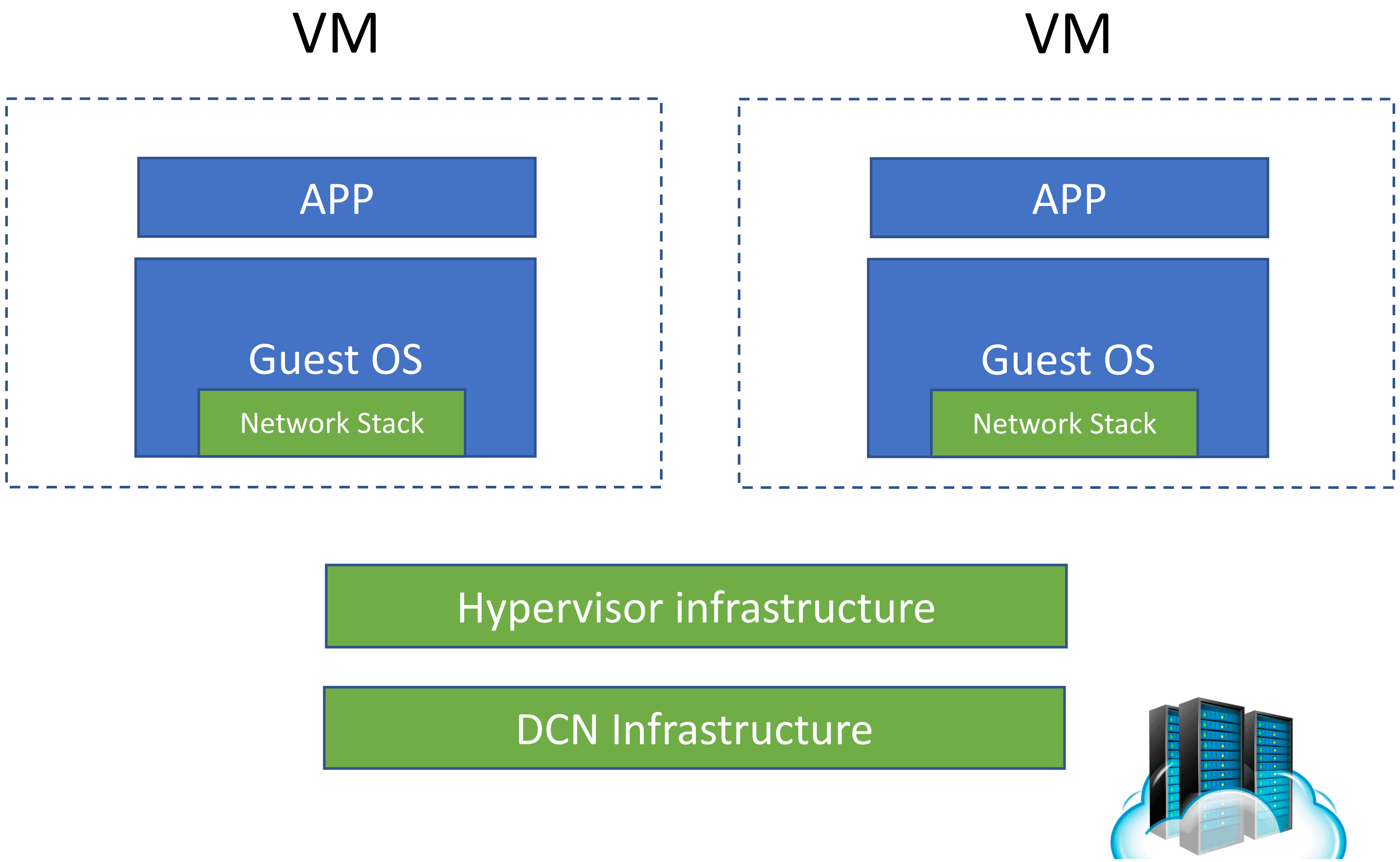
NetKernel: Making Network Stack Part of the Virtualized Infrastructure

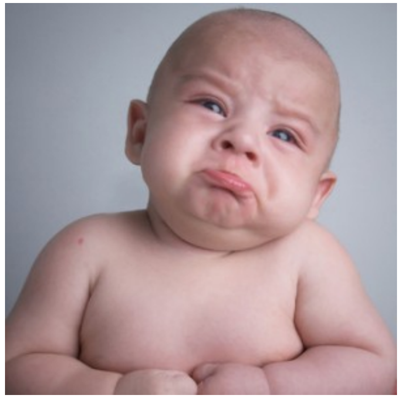
Zhixiong Niu, Hong Xu, Peng Cheng, Qiang Su, Yongqiang Xiong,
Tao Wang, Dongsu Han, Keith Winstein

Current architecture in the cloud



What’re the fundamental limitations?

Motivation: Tenants



Have to deal with the network stack all by myself

TCP parameters
initcwnd
initialRTO (ms)
minRTO (ms)
DelayedAckTimeout (ms)

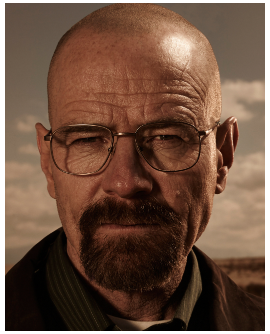
Buffer
net.ipv4.tcp_rmem
net.ipv4.tcp_wmem
net.core.rmem_max
net.core.wmem_max

BBR
MPTCP
CTCP
CUBIC
PCC
DCTCP
StackMap
MegaPipe
mTCP
FastSocket
FlexSC
Kernel

Tenants are primarily concerned with performance and functionality, not implementation details.

Motivation: Operator

I know everything here.
I can really help my tenants (and make some money!)



Resources

Kernel

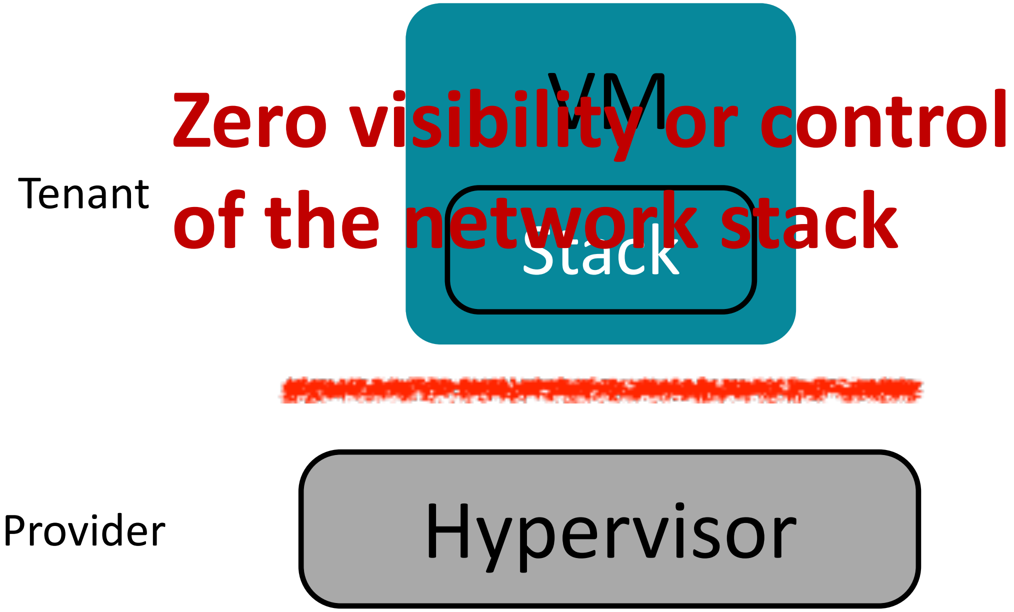
RDMA

FPGA

NIC

DPDK

Motivation: Operator



Can't deploy new stacks (DCTCP)

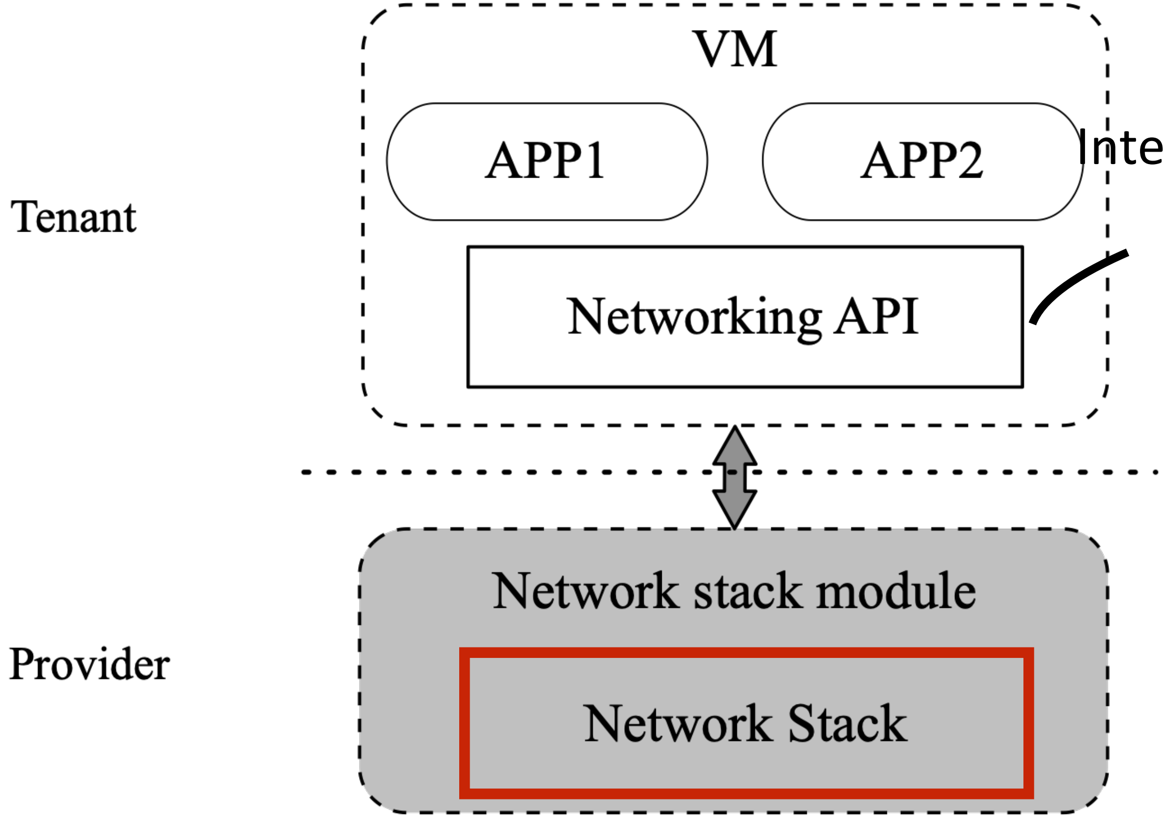
Difficult to even define performance SLA

Difficult to perform management tasks

Difficult to troubleshoot

Is there a better way?

Making Network Stack Part of the Virtualized Infrastructure



Interface unchanged (BSD sockets, etc.)

Packets handled in the NSM

Current architecture

Benefits

- Better efficiency in management for the operator
  - Orchestrate the resource provisioning strategies more flexibly
  - Implement management functions as a part of user’s network stack
- Deployment and performance gains for users without efforts
  - Enforce various kernel stack optimizations
  - Enforce high-performance userspace stacks
  - Use advanced hardware

Design Challenges

- How to transparently redirect socket API calls without changing applications?
- How to transmit the socket semantics between the VM and NSM?
- How to ensure high performance with semantics transmission (e.g., 100 Gbps)?

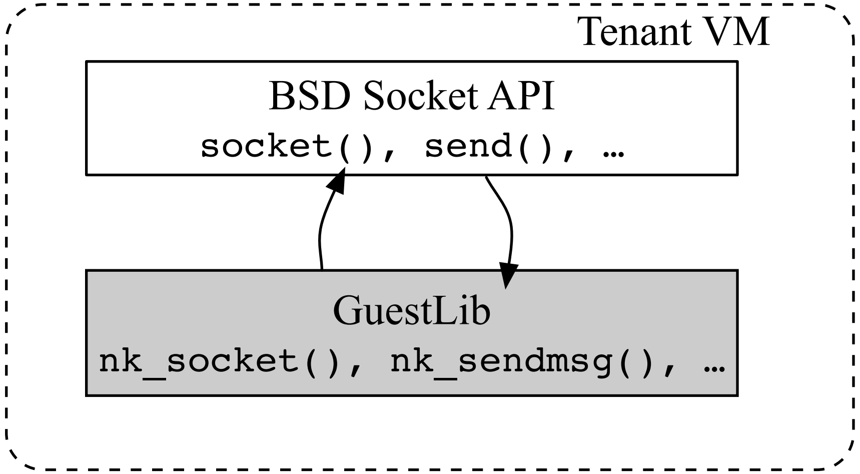
Transparent socket API redirection

- A new sock type, SOCK_NETKERNEL
- GuestLib: A complete implementation of BSD socket APIs

socket(), socket_sendmsg(), …

↓

nk_socket(), nk_socket_sendmsg(), …



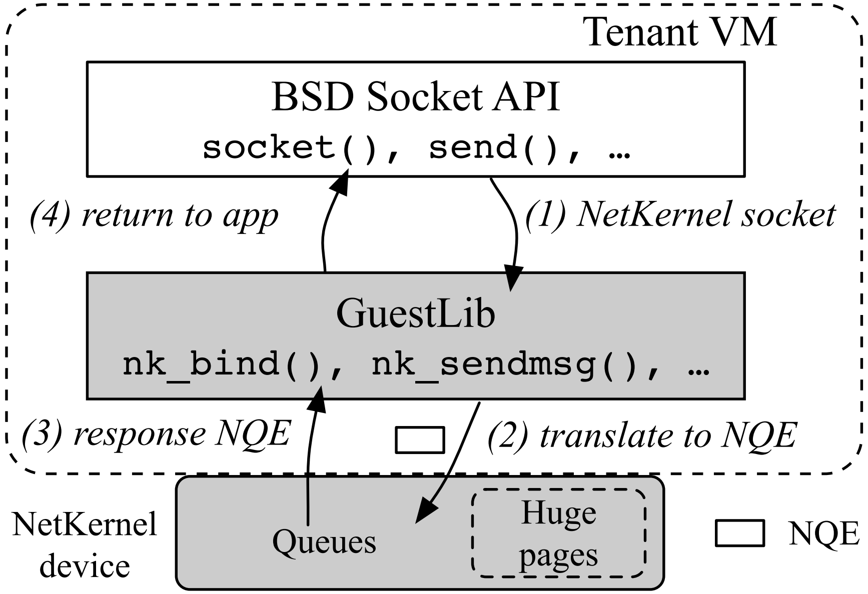
GuestLib

A lightweight semantics channel

- NQE: NetKernel queue elements for semantics

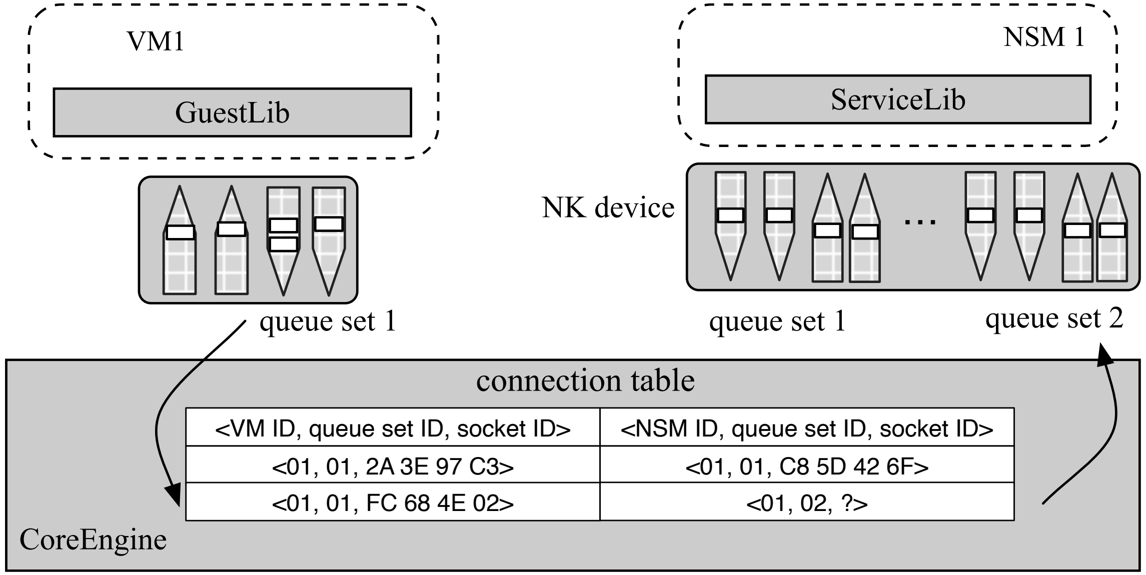
1B	1B	1B	4B	8B	8B	4B	5B
op type	VM ID	Queue set ID	VM socket ID	op_data	data pointer	size	rsved

- NQE queues for semantics transmission and hugepages for data transmission in NetKernel device



Scalable lockless queues

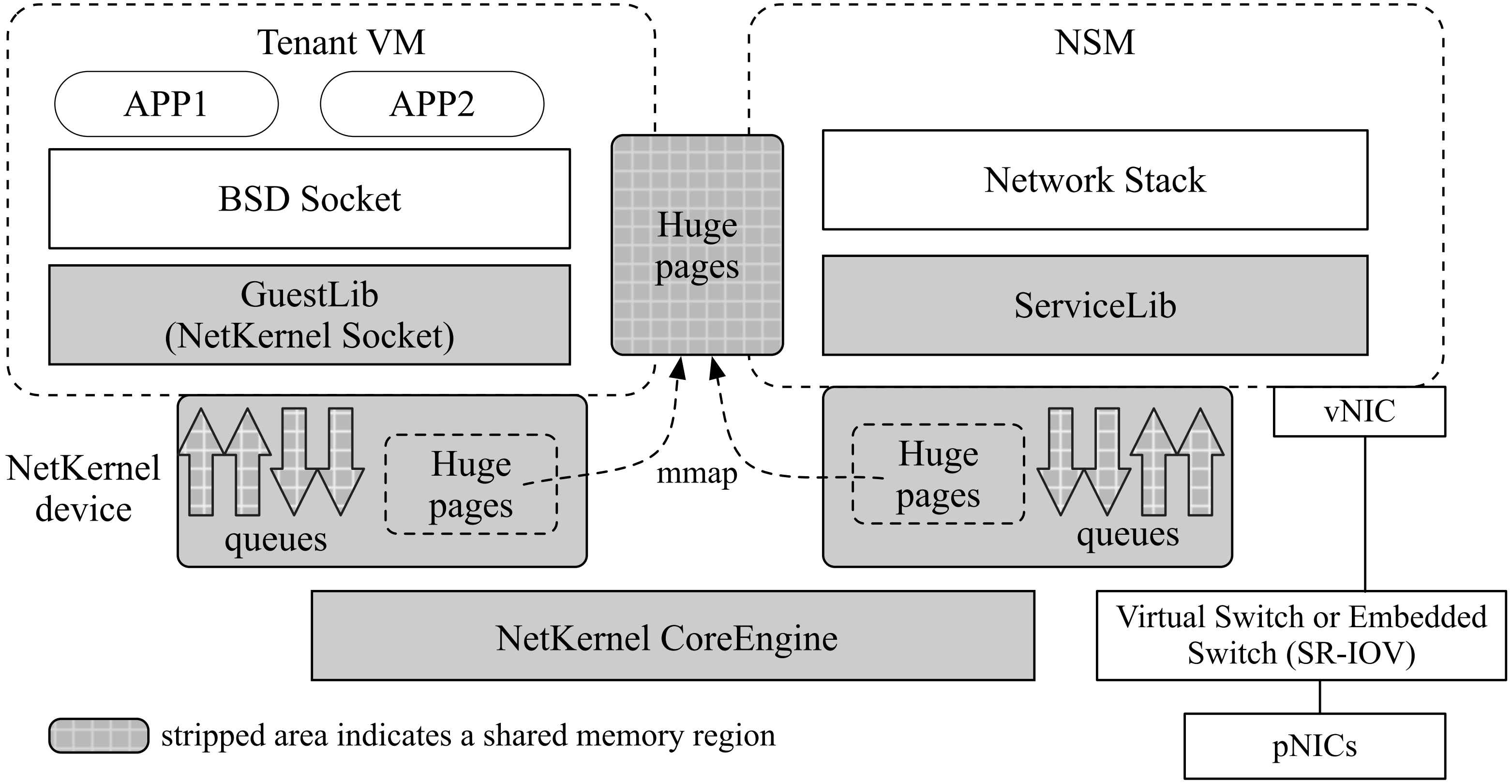
- Per-core queue set, lockless queues
- NQE switching via CoreEngine



VM based NSM.

- Supports existing kernel and userspace stacks from various Oses
- Provide good isolation to guarantee the performance
- Run stacks independent of the hypervisor

NetKernel


Tenant VM
APP1
APP2
BSD Socket
GuestLib
(NetKernel Socket)
Huge
pages
NSM
Network Stack
ServiceLib
NetKernel
device
queues
Huge
pages
mmap
Huge
pages
queues
vNIC
NetKernel CoreEngine
Virtual Switch or Embedded
Switch (SR-IOV)
pNICs
stripped area indicates a shared memory region

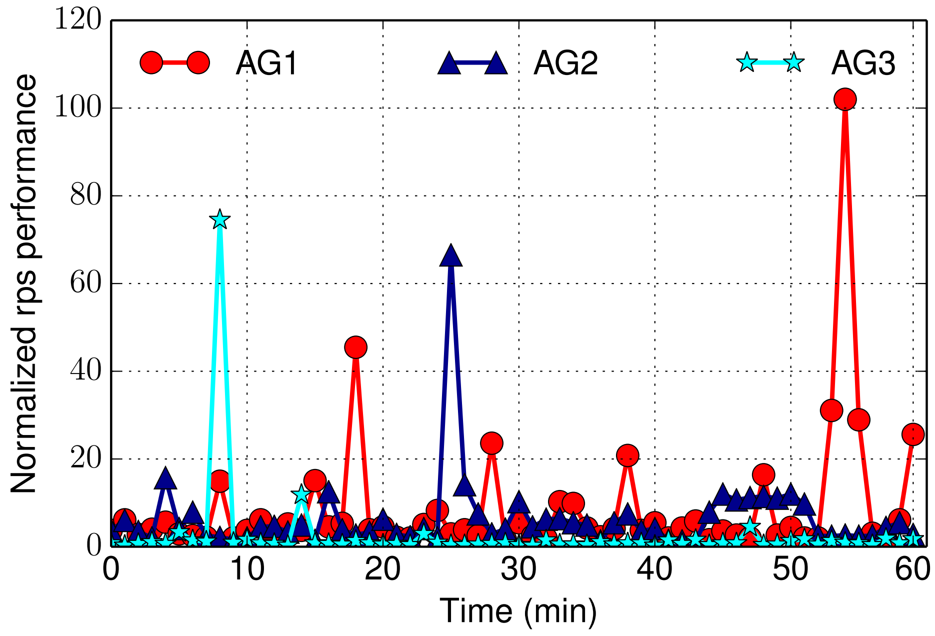
Implementation

- QEMU KVM 2.5.0, Linux Kernel 4.9
- Intel(R) Xeon(R) 16-core CPU @ 2.30GHz x 2
- 256GB DDR4 2133MHz
- Mellanox ConnectX-4 100G single port NIC

Use Cases #1: Multiplexing

Application Gateway (AG): L7 proxy and load balancing services

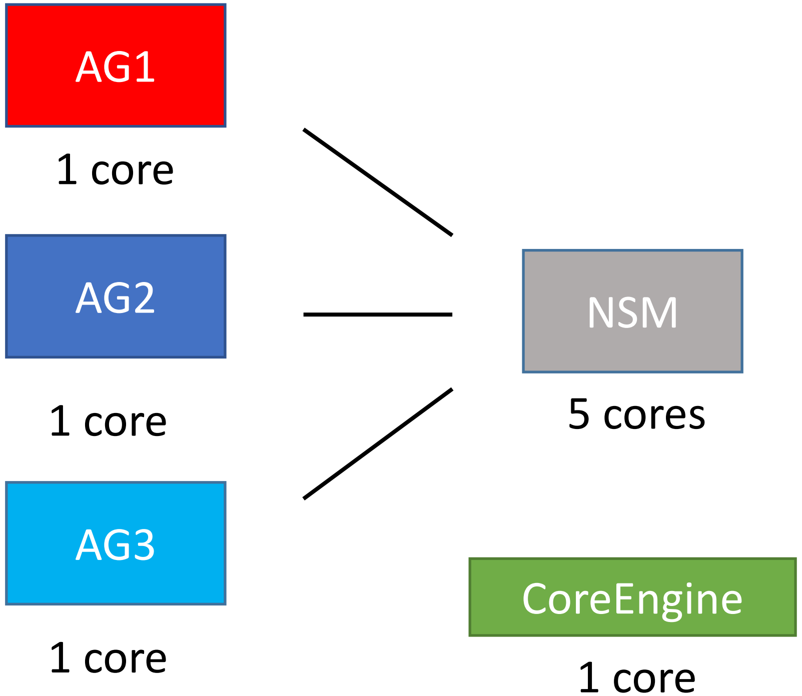
AG1	AG2	AG3
4 core	4 core	4 core

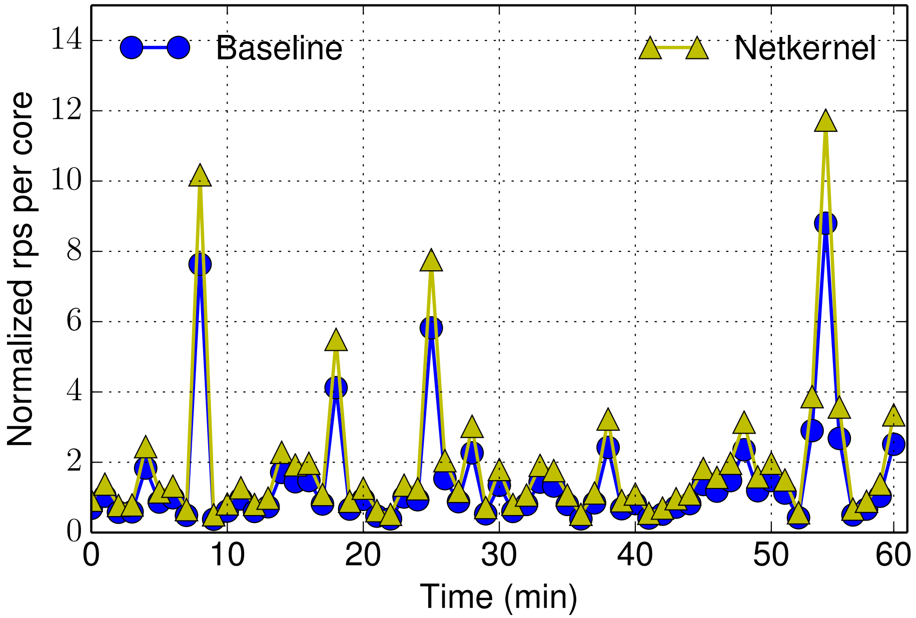


Normalized RPS Performance of a trace from a large cloud

Use Cases #1: Multiplexing

NetKernel: 9 Cores
Baseline: 12 Cores

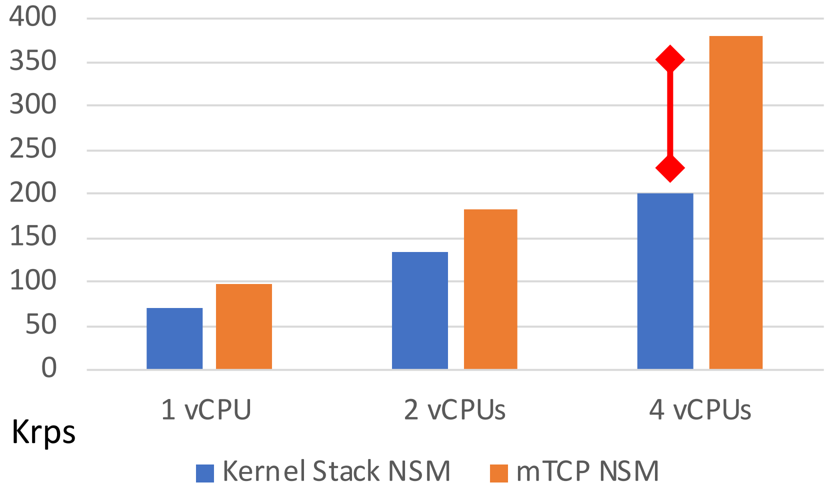




Benefit: NetKernel can help operator perform network management more efficiently

Use Case #2: Deploying mTCP without API Change

- mTCP doesn't support Nginx yet
- mTCP ported as an NSM, fixed a bug in DPDK mlx5_core driver
- Unmodified Nginx on mTCP without any tenant effort



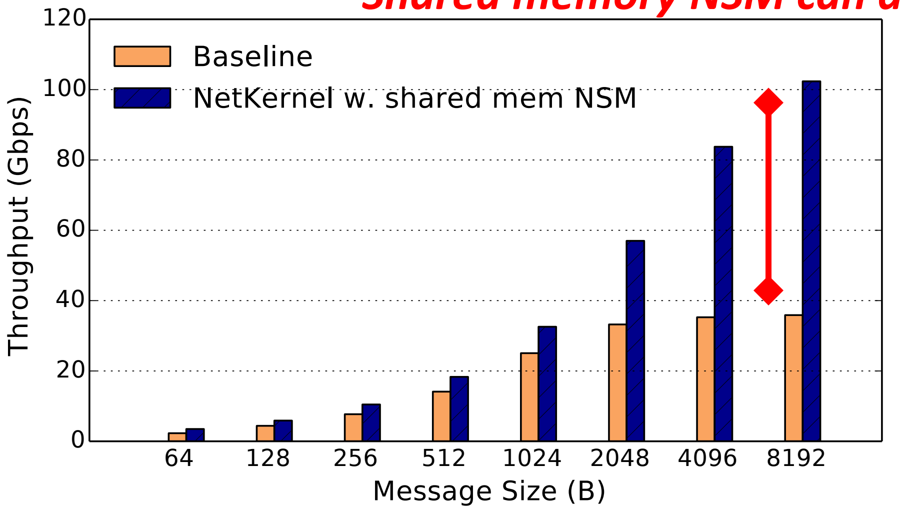
mTCP NSM brings ~1.8x performance gain

Use Case #3: Shared Memory Networking

- The operator can easily detect the on-host traffic with NetKernel
- For on-host traffic, it can use shared memory NSM to avoid TCP and bridge overhead

Deployment and performance gains for users

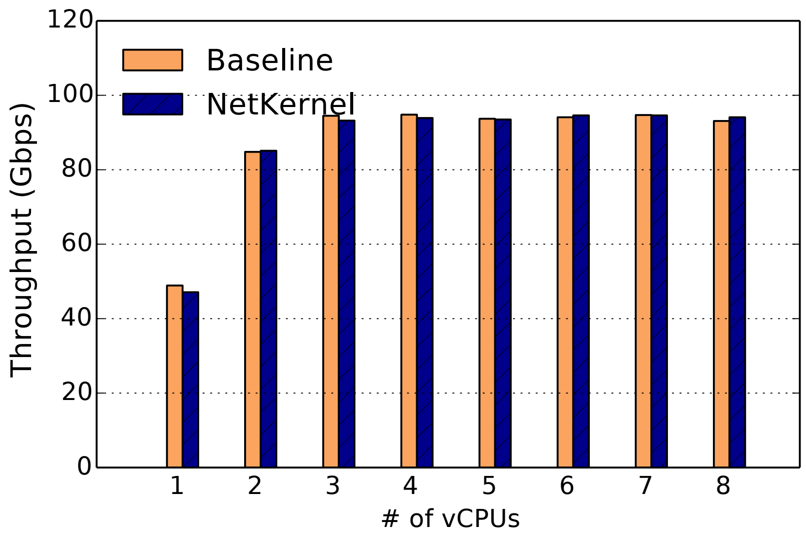
Shared memory NSM can achieve >2x performance gain for on-host traffic



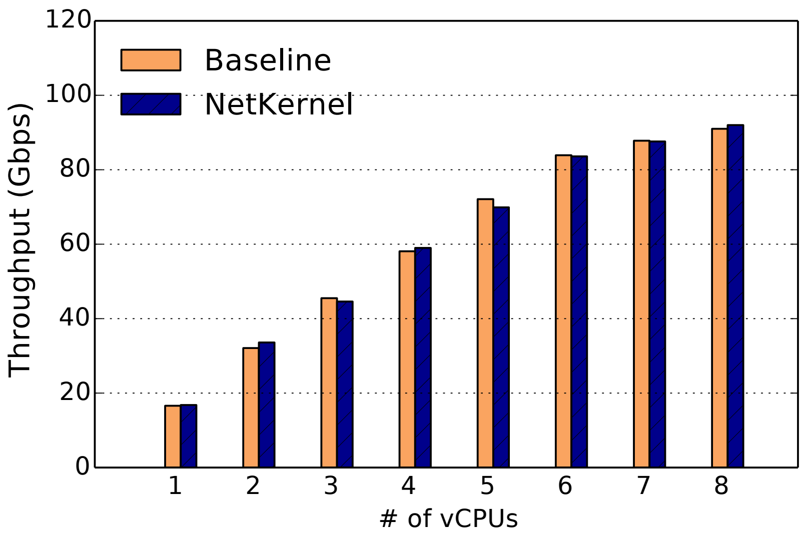
Benefit: NetKernel can help user achieve deployment and performance gains

Microbenchmarks: Throughput

- Baseline (a VM) and NetKernel (a VM with a Linux Kernel) using the same setting
- 8 TCP connections, 8KB messages



Send

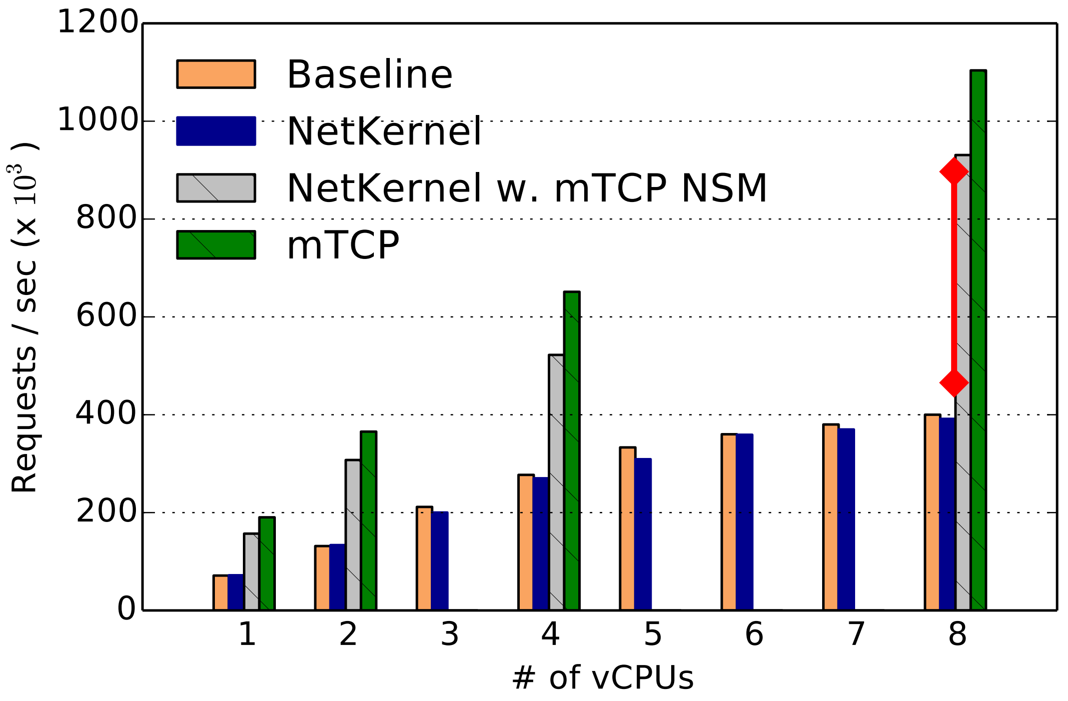


Receive

Can achieve 100Gbps with 3 cores (send), 8 cores (receive)

Microbenchmarks: RPS

- Simple epoll server, short TCP conn.
- 64B request/response



mTCP NSM brings 2x performance gain

Discussion and future directions

- How can I do Netfilter?
  - Hard to support for multiple-tenant NSM
- What about troubleshooting performance issues?
  - Operator can easily monitor their NSMs by deploy additional mechanisms in the NSMs
- Does NetKernel increase the attack surface?
  - Own address spaces for NK device
  - Isolated channel between NSM and VM
- Future directions
  - Performance isolation
  - Charging policies
  - FPGA/SoC

Recap

- Designed and implemented NetKernel
  - Decouples the network stack from the guest
  - Making it part of the virtualized infrastructure in the cloud
- Enabled several new usecases
  - Multiplexing, mTCP NSM, Shared mem. NSM
- Conducted comprehensive testbed evaluation with commodity 100G NICs
- Website
  - https://netkernel.net