Research Faculty Summit 2018
Systems | Fueling future disruptions
Accelerated Networking in Azure

Sambhrama Mundkur
Principal Software Engineer, Azure Host Networking Group
Agenda

• Azure Background
• What does scale mean in cloud?
• SDN in Azure
• Challenges in Virtualization
• Scaling SDN with SmartNIC
• Conclusion
Compute Instances

Azure Storage

Datacenter Network

100K

10’s of PB

10’s of Tbps

2010

Million S

Exabytes

Pbps

2017

Research Faculty Summit 2018
Systems | Fueling future disruptions
<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortune 500 using Microsoft Cloud</td>
<td>&gt;85%</td>
</tr>
<tr>
<td>New Azure customers a month</td>
<td>&gt;120,000</td>
</tr>
<tr>
<td>Azure storage objects</td>
<td>&gt;60 TRILLION</td>
</tr>
<tr>
<td>Azure VMs are Linux VMs</td>
<td>&gt;900 TRILLION requests/day</td>
</tr>
<tr>
<td>Azure Active Directory Orgs</td>
<td>&gt;9 MILLION</td>
</tr>
<tr>
<td>Azure Active Directory authentications/week</td>
<td>&gt;18 BILLION</td>
</tr>
<tr>
<td>Azure Event Hubs events/week</td>
<td>&gt;3 TRILLION</td>
</tr>
<tr>
<td>Azure DB requests/day</td>
<td>&gt;110 BILLION</td>
</tr>
</tbody>
</table>

**Azure Scale & Momentum**

Research Faculty Summit 2018
Systems | Fueled future disruptions
Other ways to think about scale

- Will have device failures, link failures, server failures all the time, multiple at a time
- Will have gray failures as well
  - Lossy links, switches dropping packets greater than x bytes, undetected corruption, etc.
- Must build great automation and highly available designs to detect and repair/remove such failures from the network, and you will still have ones your automation doesn’t find
- IaaS customers (e.g. Enterprises) are not tolerant of single VM failures – they expect 4-5 9s of availability
- Must be able to service all parts of the network (including the host) and still achieve this availability
- Performance, availability, serviceability downtimes, are all measured by P99/P99.9, not P50
SDN: Building the right abstractions for Scale

Abstract by separating management, control, and data planes

Example: ACLs

<table>
<thead>
<tr>
<th>Management Plane</th>
<th>Create a tenant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Plane</td>
<td>Plumb these tenant ACLs to these switches</td>
</tr>
<tr>
<td>Data Plane</td>
<td>Apply these ACLs to these flows</td>
</tr>
</tbody>
</table>

Data plane needs to apply per-flow policy to millions of VMs

How do we apply billions of flow policy actions to packets?
Virtual Filtering Platform (VFP)  
Azure’s SDN Dataplane

- Acts as a virtual switch inside Hyper-V VMSwitch
- Provides core SDN functionality for Azure networking services, including:
  - Address Virtualization for VNET
  - VIP -> DIP Translation for SLB
  - ACLs, Metering, and Security Guards
- Uses programmable rule/flow tables to perform per-packet actions
- Supports all Azure dataplane policy at 40GbE+ with offloads
Flow tables: The right abstraction for the host

VSwitch exposes a typed Match-Action-Table API to the controller
Controllers define policy
One table per policy

Key insight: Let controller tell switch exactly what to do with which packets
e.g. encap/decap, rather than trying to use existing abstractions (tunnels, ...)

---

**VFP**

- **TO: 10.2/16** Encap to GW
- **TO: 10.1.1.5** Encap to 10.5.1.7
- **TO: 110/8** NAT out of VNET

---

**VNET**

- **TO: 10.1.1.2** DNAT to 10.1.1.2
- **TO: 110/8** SNAT to 79.3.1.2

---

**LB NAT**

- **TO: 10.1.1/24** Allow
- **10.4/16** Block
- **TO: 110/8** Allow

---

**ACLS**

- **TO: 79.3.1.2** DNAT to 10.1.1.2
- **TO: 10.1.1/24** Allow
- **10.4/16** Block
- **TO: 110/8** Allow

---

**VM 1 10.1.1.2**
Host SDN Scale Challenges in Practice

- Hosts are Scaling Up: 1G → 10G → 40G → 50G → 100G
  - Reduces COGS of VMs (more VMs per host) and enables new workloads
  - Need the performance of hardware to implement policy without CPU
  - Not enough to just accelerate to ASICs – need to move entire stacks to HW

- Need to support new scenarios: BYO IP, BYO Topology, BYO Appliance
  - We are always pushing richer semantics to virtual networks
  - Need the programmability of software to be agile and future-proof—12-18 month ASIC cycle + time to roll new HW is too slow

- How do we get the performance of hardware with programmability of software?
Our Solution: Azure SmartNIC (FPGA)

- HW is needed for scale, perf, and COGS at 40G+
- 12-18 month ASIC cycle + time to roll new HW is too slow
- To compete and react to new needs, we need agility—SDN
- Programmed using Generic Flow Tables
  - Language for programming SDN to hardware
  - Uses connections and structured actions as primitives
Azure Accelerated Networking: Fastest Cloud Network!

• Highest bandwidth VMs of any cloud
  • Standard compute (D series) VMs get 25Gbps
  • Big compute (M series) gets 32Gbps
  • Standard Linux VM with CUBIC gets 30+Gbps on a single connection

• Consistent low latency network performance
  • Provides SR-IOV to the VM
  • Up to 10x latency improvement – sub 25us within VM Scale Sets
  • Increased packets per second (PPS)
  • Reduced jitter means more consistency in workloads

• Enables workloads requiring native performance to run in cloud VMs
  • >2x improvement for many DB and OLTP applications
Accelerated Networking Internals

SDN/Networking policy applied in software FPGA acceleration used to apply all policies in the host
SmartNIC—Accelerating SDN

Controller

Controller

Controller

SLB Decap

SLB NAT

VNET

ACL

Metering

Transposition Engine

GFT Offload Engine

VFP

VMSwitch

SmartNIC

GFT

Crypto

RDMA

QoS

First Packet

SR-IOV (Host Bypass)

VM

ARM APIs

VFP APIs

GFT Offload API (NDIS)

50G
Serviceability is Key

• All parts of this system can be updated, any of which require us to take out the hardware path
  • FPGA image, driver, GFT layer, Vswitch/VFP, NIC PF driver
• IaaS requires high uptime and low disruption —can’t take away the NIC device from under the app, and can’t reboot the VM / app
• Instead, we keep the synthetic vNIC and support transparent failover between the vNIC and VF

Lesson: A huge amount of the effort to deploy SR-IOV was in making all parts of this path rebootlessly serviceable without impact
Lessons Learnt

• Design for serviceability upfront
• Use software development techniques for FPGAs
• Better perf means better reliability
• HW/SW co-design is best when iterative
• Failure rates remain low
• Upper layers should be agnostic of offloads
Questions?
Thank you