Microsoft Azure

Azure DPDK Performance Report

April 2025

Microsoft Azure Network Adapter (MANA)

VM Size: Standard_E192ids_v6

Prepared by:
Matthew G. McGovern
Microsoft Linux Systems Group

a

Microsoft Azure

Index

Purpose

The purpose of this document is to share basic performance data for DPDK on Microsoft Azure. The document covers Linux Virtual Machines (VMs) with the Microsoft Azure Network Adapter (MANA) on the Standard_E192ids_v6 offering. The document outlines the test environment, test cases and test data; as well as tools to reproduce the results.

Target Audience

The intended audience of this document are software developers interested in evaluating the performance of DPDK on Azure. The reader should be familiar with the Data Plane Development Kit (DPDK), operating system (OS) virtualization and Linux.

Azure Test Environment Description:

Linux Integration Services Automation (LISA)

Microsoft's automated Azure testing for Linux leverages an open-source framework maintained by Microsoft: Linux Integration Services Automation (LISA). Knowledge of LISA is not required to reproduce the test results. A repository with shell scripts to prepare an environment and run the tests are available on Github.

Dedicated Network Adapter for DPDK

DPDK test cases on Azure require at least one dedicated network device; we recommend leaving one management NIC available while using DPDK on Azure. The tests in this report leave eth0 open and use eth1 for DPDK.

Note on Physical Proximity

The physical location of a resource in Azure is normally determined by the 'region' or 'location' parameter. This location parameter corresponds to specific datacenters which vary in size. Customers can leverage the concept of 'availability sets' (a failover and data proximity abstraction in Azure) for physical proximity within a server rack. The test results in this report were recorded without leveraging any options to guarantee physical proximity. However, the VM size Standard_E192ids_v6 is guaranteed to ensure a single VM per server blade. The test VMs were created in the same Azure region within a few months of the release of the Standard_E192ids_v6 size. Some amount of physical proximity was a side effect of using this new size within a single region.

VM Size and Capabilities

- VM SKU: Standard_E192ids_v6
- Cores: 192
- Memory: 1832GB
- Maximum NICs: 8
- Max Network Bandwidth: 200000 Mb/s
- The full specifications for this size are available here.

Performance Metrics

The testpmd application measures sent, received and dropped packets. The application displays the data in packets-per-second (PPS). Packet size was set to the default value of 64 bytes.

VM Isolation

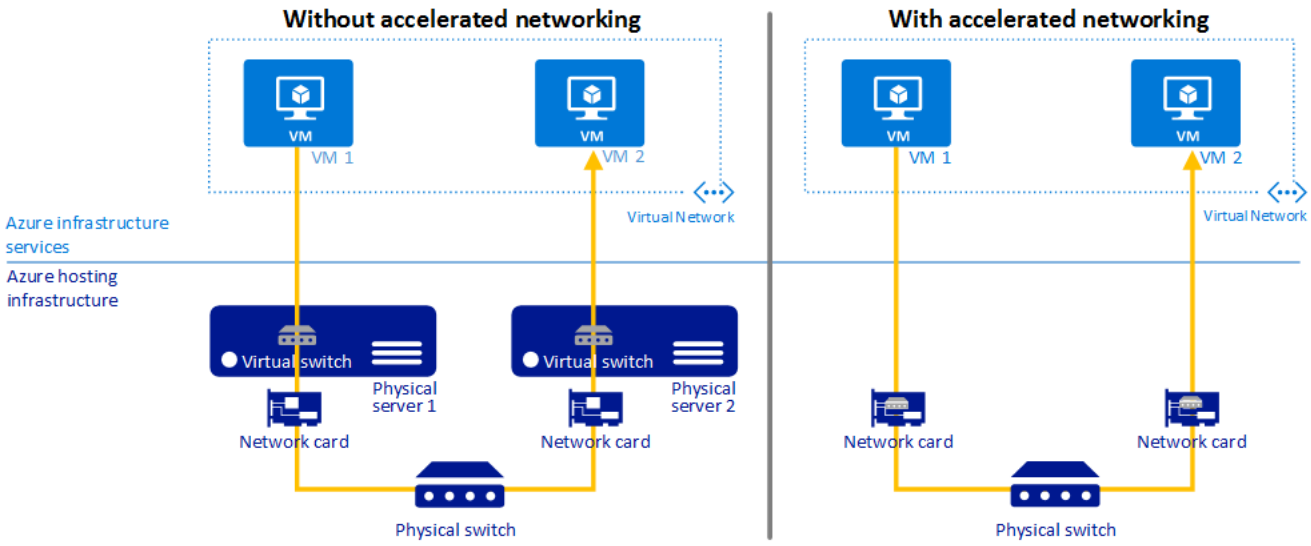
Standard_E192ids_v6 is an isolated VM size, it will never share a host with another VM. These PPS results are specific to this size. PPS tests on other VM families and VM sizes may result in lower PPS measurements. Maximum expected packet-per-second counts are hardware dependent and may change depending on the hardware and software that hosts a given VM.

Microsoft Azure

Hardware Descriptions:

Azure Accelerated Networking and SR-IOV

Azure VMs can directly manage the hardware queues on a physical NIC via Single Route I/O Virtualization (SR-IOV). The result is higher throughput and lower latency than would be possible for an entirely virtualized network interface. Accelerated Networking was previously an optional feature for Azure VMs. Newer generations of Azure VM sizes require SR-IOV, so Accelerated Networking is always enabled for Standard_E192ids_v6 NICs.



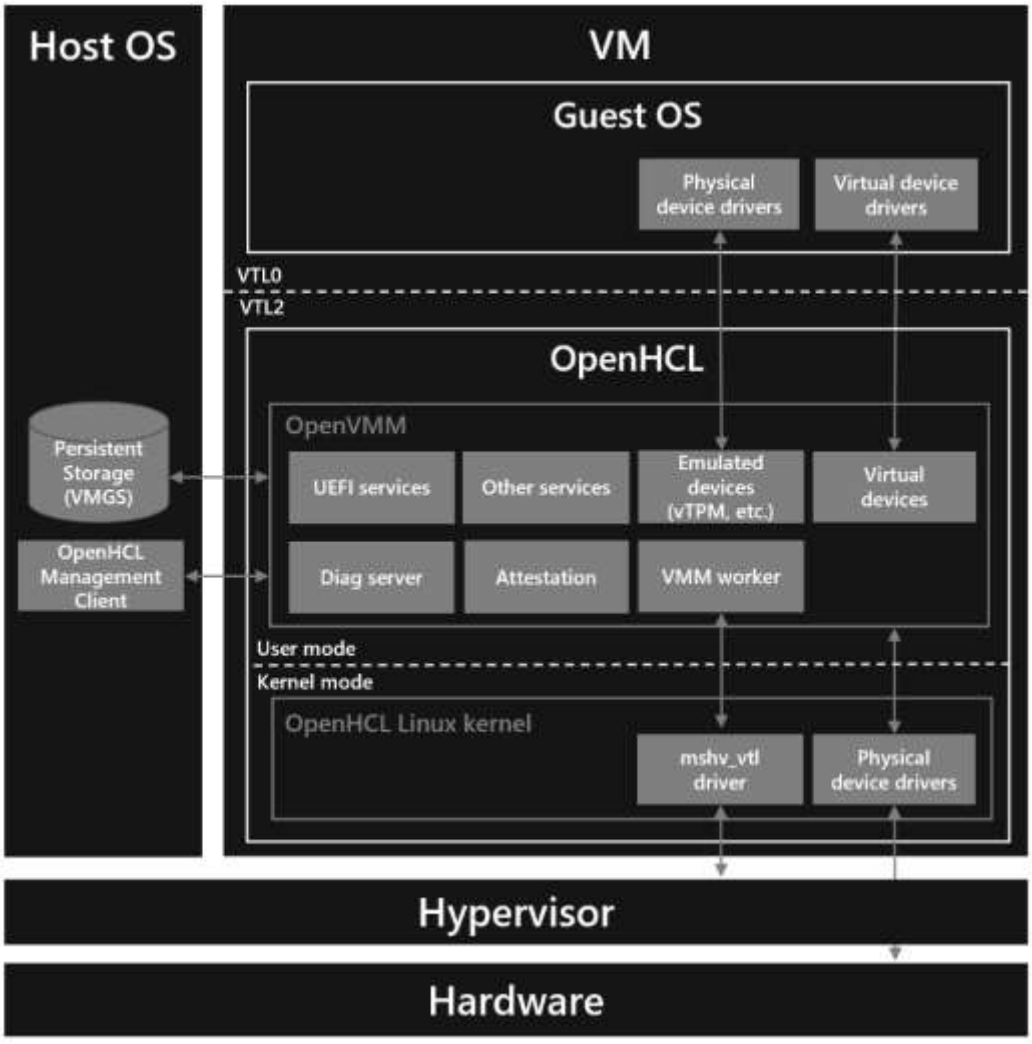
https://learn.microsoft.com/en-us/azure/virtual-network/accelerated-networking-overview

OpenHCL

Azure Standard_E192ids_v6 VMs use an updated virtualization model compared to other offerings on Azure. Many services which required interaction with the Windows Host OS partition are now offloaded to an OpenHCL paravisor running within the guest partition. This change enables security and reliability features which are out-of-scope for this document.

Interested readers can find a high-level description on the Microsoft Tech Blog: OpenHCL.

Both OpenHCL and OpenVMM are open source: microsoft/openvmm on Github



OpenHCL: the new, open source paravisor | Microsoft Community Hub

CPU Layout and Non-Uniform Memory Access (NUMA):

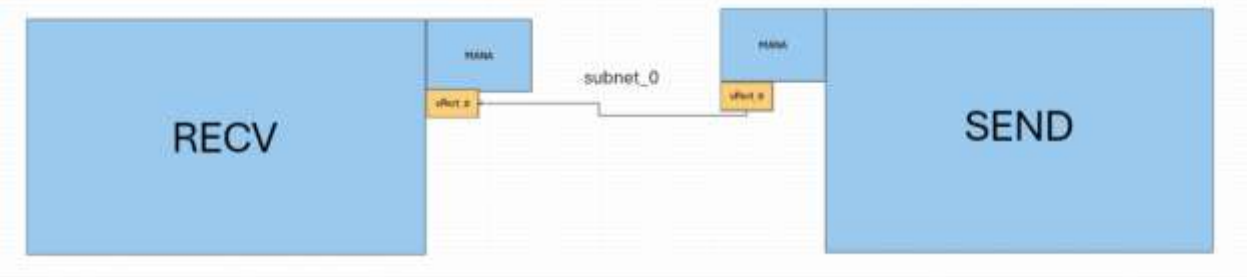
This report provides data for the MANA NIC. The data is from a Standard_E192ids_v6 VM. This size uses an Intel 5th Generation Xeon (Emerald Rapids) CPU. Helpful information about the NUMA layout, CPU cores, caches and threads-per-core can be found with the lscpu application or the DPDK tool dpdk/usertools/cpu_layout.py. Note that attempting to use multiple threads on the same physical core can degrade performance. Our tests aim to assign one DPDK lcore to each physical CPU by omitting certain core indices from the testpmd core list argument.

lscpu

```
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          48 bits physical, 57 bits virtual
  Byte Order:             Little Endian
CPU(s):                   192
  On-line CPU(s) list:    0-191
Vendor ID:                GenuineIntel
  Model name:             INTEL(R) XEON(R) PLATINUM 8573C
    CPU family:           6
    Model:                207
    Thread(s) per core:   2
    Core(s) per socket:   48
    Socket(s):            2
    Stepping:             2
    CPU max MHz:          2300.0000
    CPU min MHz:          800.0000
    BogoMIPS:             4599.99
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge m
                          ca cmov pat pse36 clflush mmx fxsr sse sse2 ss ht sysc
                          all nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xt
                          opology tsc_reliable nonstop_tsc cpuid aperfmperf pni
                          pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic
                           movbe popcnt tsc_deadline_timer aes xsave avx f16c rd
                          rand hypervisor lahf_lm abm 3dnowprefetch tpr_shadow e
                          pt vpid ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep
                          bmi2 erms invpcid rtm avx512f avx512dq rdseed adx smap
                           avx512ifma clflushopt clwb avx512cd sha_ni avx512bw a
                          vx512vl xsaveopt xsavec xgetbv1 xsaves user_shstk avx_
                          vnni avx512_bf16 vnmi avx512vbmi umip waitpkg avx512_v
                          bmi2 gfni vaes vpclmulqdq avx512_vnni avx512_bitalg av
                          x512_vpopcntdq la57 rdpid cldemote movdiri movdir64b f
                          srm serialize tsxldtrk ibt amx_bf16 avx512_fp16 amx_ti
                          le amx_int8 arch_capabilities
Virtualization features:
  Virtualization:         VT-x
  Hypervisor vendor:      Microsoft
  Virtualization type:    full
Caches (sum of all):
  L1d:                    4.5 MiB (96 instances)
  L1i:                    3 MiB (96 instances)
  L2:                     192 MiB (96 instances)
  L3:                     520 MiB (2 instances)
NUMA:
  NUMA node(s):           4
  NUMA node0 CPU(s):      0-47
  NUMA node1 CPU(s):      48-95
  NUMA node2 CPU(s):      96-143
  NUMA node3 CPU(s):      144-191
```

Test Description:

dpdk-testpmd:



A Sender VM sends traffic to a Receiver VM using dpdk-testpmd. The Sender VM sets the testpmd `--tx-ip` flag to the destination address. Packet size is left as the default (64 bytes, UDP).

testpmd invocation notes:

Our tests use the testpmd --tx-ip flag to define the IP address of the source and destination vNICs. This flag is used because Azure Virtual Networks (vNets) will not route based on the MAC address alone. The vNet configuration and any associated subnet and virtual routing table rules must be configured for traffic to flow. Packets generated by testpmd must have an IP address and port number that satisfies the vNet configuration rules to successfully traverse the vNet, and those rules are defined for layer 3.

The --txonly-multi-flow flag is enabled to spread traffic across multiple UDP ports; this ensures that there is enough entropy in the packet UDP headers for receive-side-scaling (RSS) to hash the packets and spread them across multiple queues.

Microsoft Azure

testpmd send/receive with multiple queues

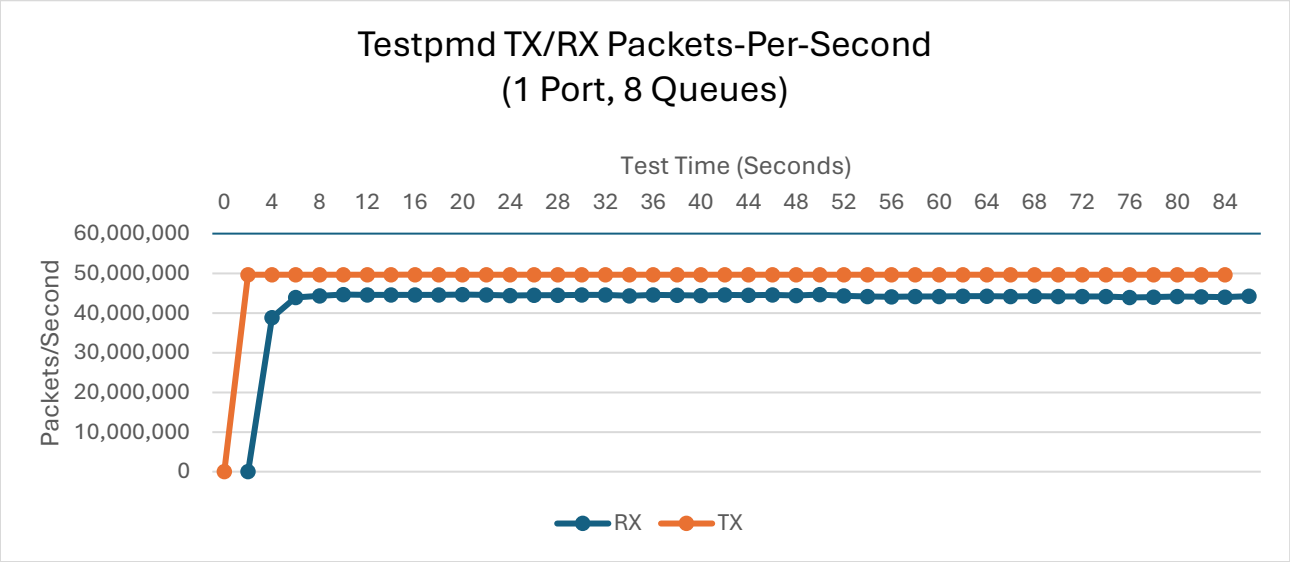
Forwarding vCPUs: 8 sending, 16 receiving
Hugepages Per NUMA node (2MB): 4096
Network Adapter: Microsoft Azure Network Adapter
CPU Type: Intel 5th Generation Xeon (Emerald Rapids)
Azure VM SKU: Standard_E192ids_v6
Hyper-V Version: 10.0.26100.1145-1-0
Linux Kernel Version: 6.8.0-1021-azure
Azure Marketplace Image: Canonical Ubuntu-24_04-lts server 24.04.202502210
RDMA Core Version: 52.2
DPDK Version: v24.11
DPDK Queue Count: 4 sender, 8 receiver
Tools to reproduce: dpdk-perf: DPDK performance report data and scripts
Application invocations:

```
#Receiver:
sudo dpdk-testpmd -l 3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47 --
vdev="7870:00:00.0,mac=7c:1e:52:0f:c6:95" -- --forward-mode=rxonly --auto-start   --
txd=2048 --rxd=2048 --txq=8 --rxq=8 --stats 2  --nb-cores=16

#Sender:
dpdk-testpmd -l 3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47 --
vdev="7870:00:00.0,mac=60:45:bd:c0:eb:a3" -- --forward-mode=txonly --auto-start   --
rxd=2048 --txd=2048 --txq=4 --rxq=4 --stats 2 --tx-ip="10.0.1.4,10.0.1.5"  --txonly-
multi-flow --nb-cores=8
```

Average PPS:

- **Sender: 49665398**
- **Receiver: 44208919**



Commentary

These results show our most recent maximum measurement for packets-per-second where the receiver drops no packets. To ensure the sender generates and transmits as many packets as possible, we allow the sender to generate more packets than can be transmitted by the NIC. Note that our data shows a higher PPS count on the sender side; however, this includes packets which were dropped by the sender side NIC. The receiver PPS count is the more accurate measure of performance in our data due to this test idiosyncrasy.

We did not attempt to optimize a balance between network throughput and packets-per-second. One can verify the maximum network throughput of 200 Gbps by modifying the testpmd '--txpkts' parameter to send larger packets.

Our results measured packets-per-(1)-second; however, output was collected every two (2) seconds using the testpmd '--stats 2' flag. This accounts for the initial misalignment of the TX and RX data in our graph. Pinging between the sender and receiver in this cluster took ~2-3ms for the first packet and ~1ms afterwards.

Further reading

- Data Plane Development Kit @ dpdk.org
- DPDK MANA Poll Mode Driver Documentation
- Azure Documentation: DPDK on Linux with MANA
- Linux Kernel: mana_ib driver source
- rdma-core: mana provider
- Azure Accelerated Networking documentation

Acknowledgements:

Thanks to the DPDK Project, Linux Foundation, Microsoft Linux Systems Group, Azure Linux, Azure Boost, Azure Host Networking and Network Performance groups; as well as the OpenHCL contributors.

Additional personal thanks (in no specific order) to Khrum Kashan Jat, Long Li, Wei Hu, Haiyang Zhang, Dexuan Cui, Boqun Feng, Matt Reat, Brian Denton, Paul Rosswurm, Sharath George John, Margaret Sands, Avi Levy, Shradha Gupta, Konstantin Taranov and John Bruner for their help along the way.

Questions? Concerns? Found a bug?

dpdk@microsoft.com