EXPLORING AND OPTIMIZING SCALABILITY FOR HIGH PERFORMANCE VIRTUAL SWITCHING

Zhihong Wang – zhihong.wang@intel.com

Ian Stokes – ian.stokes@intel.com

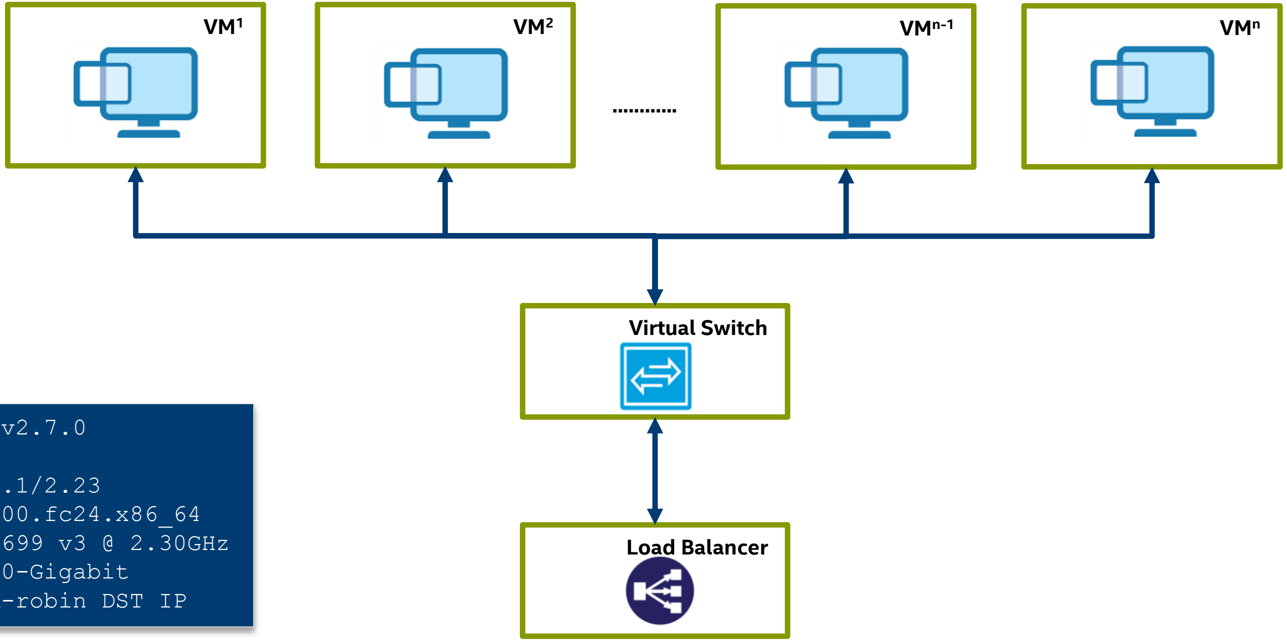
Notices and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction. Results have been estimated or simulated using internal Intel analysis or architecture simulation or modelling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

A Typical Use Case

Cloud computing multi-tenancy



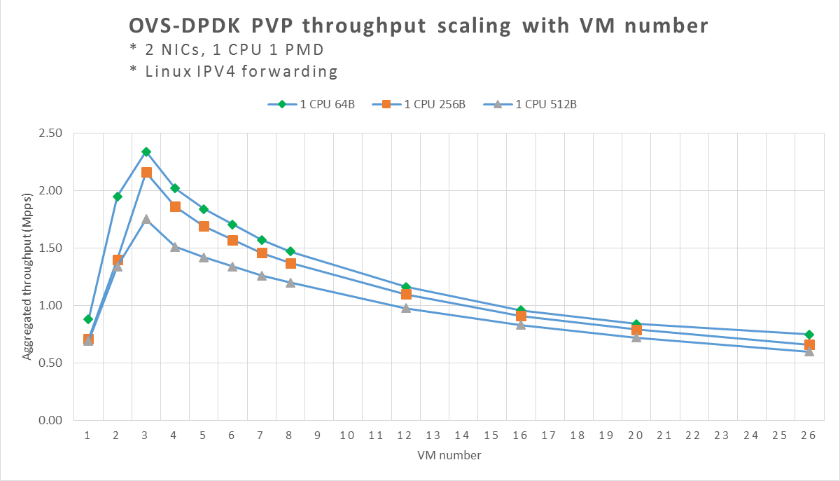
Open vSwitch: v2.7.0
DPDK: 17.02
GCC/GLIBC: 6.2.1/2.23
Linux: 4.7.5-200.fc24.x86_64
CPU: Xeon E5-2699 v3 @ 2.30GHz
NIC: 82599ES 10-Gigabit
Traffic: Round-robin DST IP

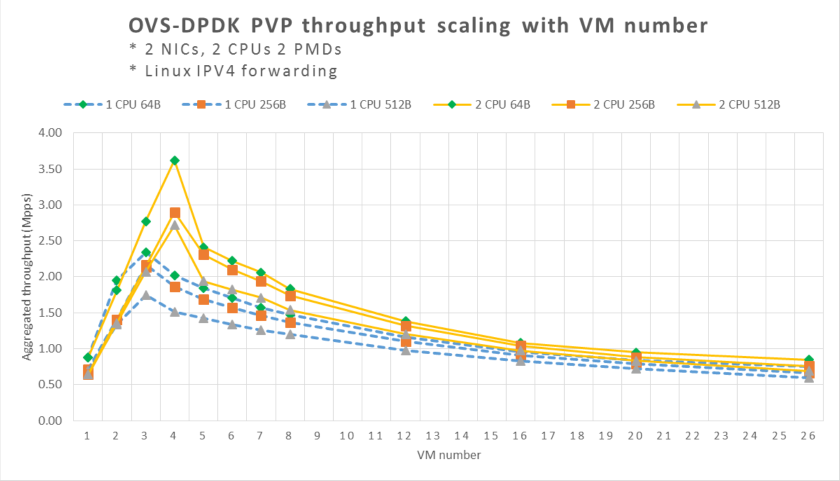
OvS-DPDK Scalability AS IS

Throughput drops significantly as the number of guests increases

Performance doesn't scale by increasing the number of CPUs

Lack of clear guidelines for customer deployment

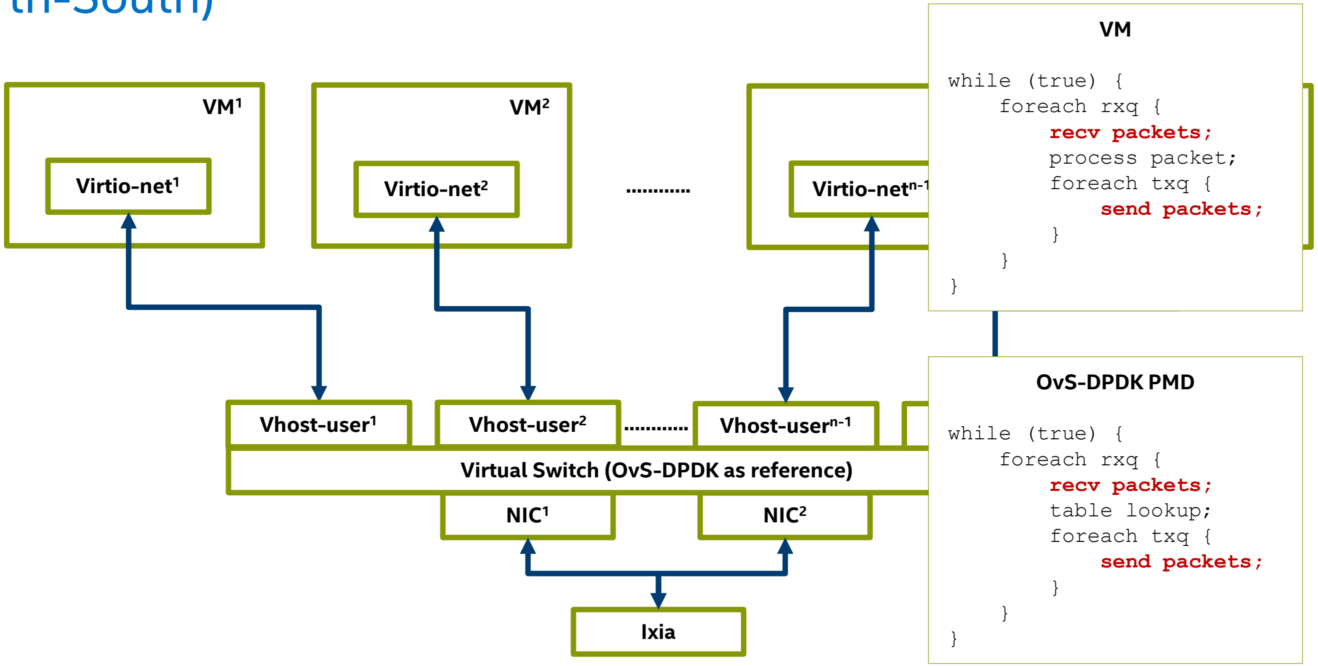




* Test and System Configurations: Estimates are based on internal Intel analysis using Intel® Server Board S2600WT, Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, Intel® 82599ES 10 Gigabit Ethernet Controller

Datapath Abstraction

PVP (North-South)



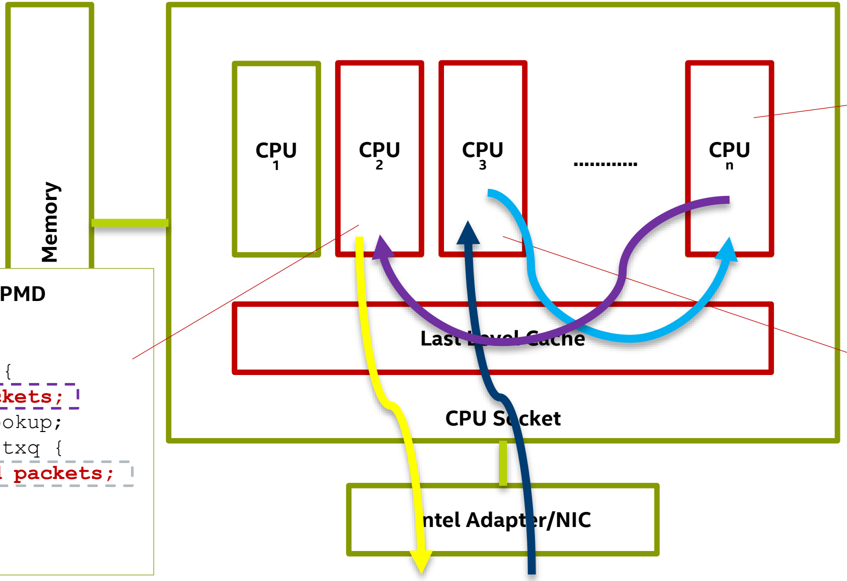

```
while (true) {
    foreach rxq {
        recv packets;
        process packet;
        foreach txq {
            send packets;
        }
    }
}
```

```
while (true) {
    foreach rxq {
        recv packets;
        table lookup;
        foreach txq {
            send packets;
        }
    }
}
```

Mapping To Hardware

VM

```
while (true) {
    foreach rxq {
        recv packets;
        process packet;
        foreach txq {
            send packets;
        }
    }
}
```



OvS-DPDK PMD

```
while (true) {
    foreach rxq {
        recv packets;
        table lookup;
        foreach txq {
            send packets;
        }
    }
}
```

OvS-DPDK PMD

```
while (true) {
    foreach rxq {
        recv packets;
        table lookup;
        foreach txq {
            send packets;
        }
    }
}
```

Analysis

Cache efficiency

CPU efficiency

CPU utilization

Work hard

Work smart

Multitasking

=

Good hardware work ethic

So-called optimization

Cache Efficiency: Maximize Burst Size

Vhost Tx Fragmentation: 1 burst for multiple guests

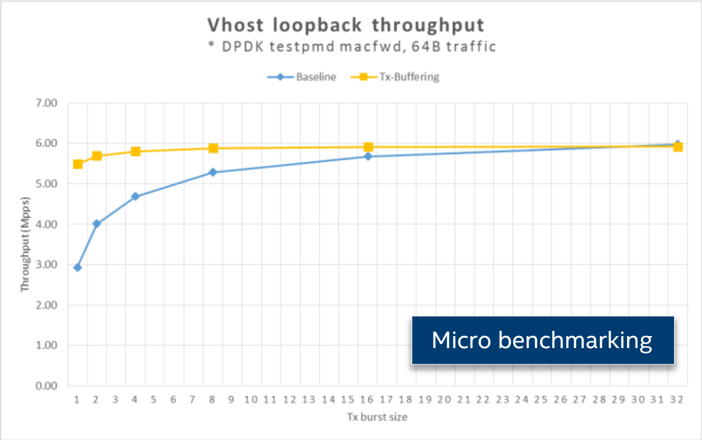
How does it impact? Poor cache efficiency!

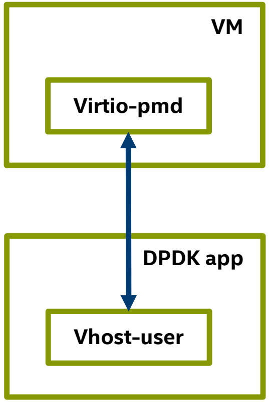
The solution: Tx-Buffering

Side effects: Increase of latency

Narrow the gap

- For throughput peaks only
- Timeout settings tuning





* Test and System Configurations: Estimates are based on internal Intel analysis using Intel® Server Board S2600WT, Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, Intel® 82599ES 10 Gigabit Ethernet Controller

CPU Efficiency: Minimize Empty Polling

Allocate CPU based on expected traffic model

- NIC cycle per 64B packet: ~30
- Vhost cycle per 64B packet: ~70 + memory latency
- NIC Rx + Vhost Tx ~= Vhost Rx + NIC Tx

```
[root]# ./ovs-appctl dpif-netdev/pmd-rxq-show
pmd thread numa_id 0 core_id 13:
        isolated : false
        port: Vhost-user1       queue-id: 0
        port: Vhost-user2       queue-id: 0
        port: Vhost-user4       queue-id: 0
pmd thread numa_id 0 core_id 12:
        isolated : false
        port: dpdk0     queue-id: 0
        port: Vhost-user3       queue-id: 0
```

The default core mapping doesn't fit for all

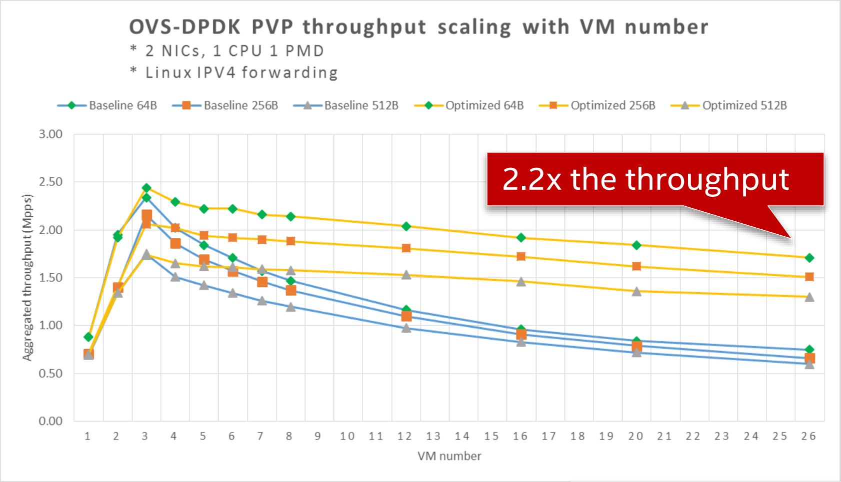
pmd-rxq-affinity

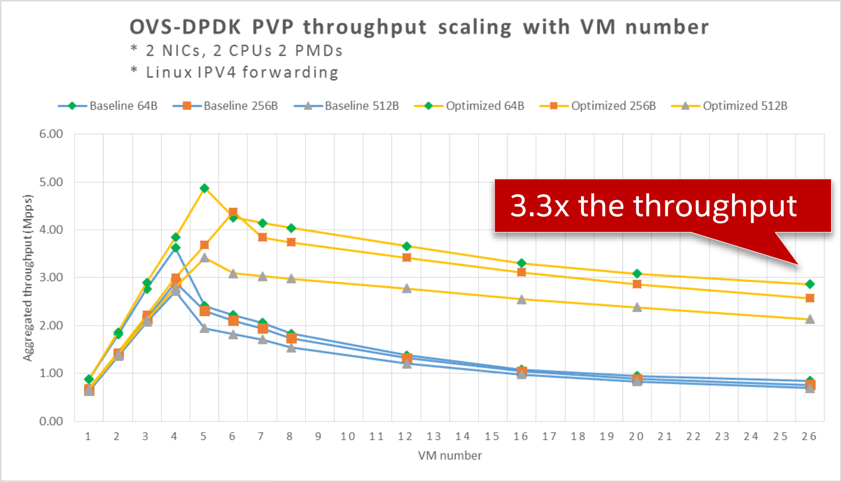
1:1 core allocation for NIC and Vhost if 1:1 north vs. south traffic

```
[root]# ./ovs-appctl dpif-netdev/pmd-rxq-show
pmd thread numa_id 0 core_id 13:
        isolated : true
        port: Vhost-user1       queue-id: 0
        port: Vhost-user2       queue-id: 0
        port: Vhost-user3       queue-id: 0
        port: Vhost-user4       queue-id: 0
pmd thread numa_id 0 core_id 12:
        isolated : true
        port: dpdk0     queue-id: 0
```

How Did We Do?

Not bad





Test based on a POC patch, OvS formal patch WIP

* Test and System Configurations: Estimates are based on internal Intel analysis using Intel® Server Board S2600WT, Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, Intel® 82599ES 10 Gigabit Ethernet Controller

CPU Utilization: Enable Hyper-Threading

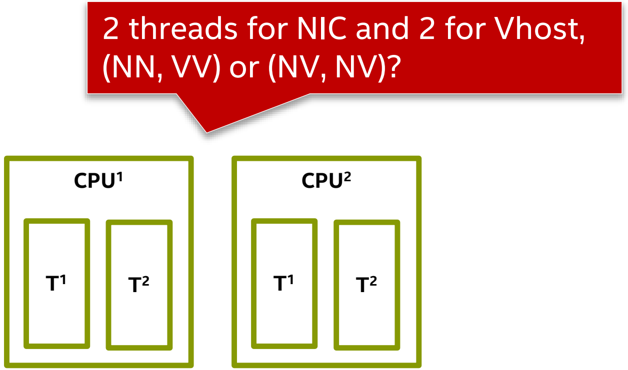
Does Hyper-Threading help? Yes!

- Hiding memory access latency
- Squeezing execution units

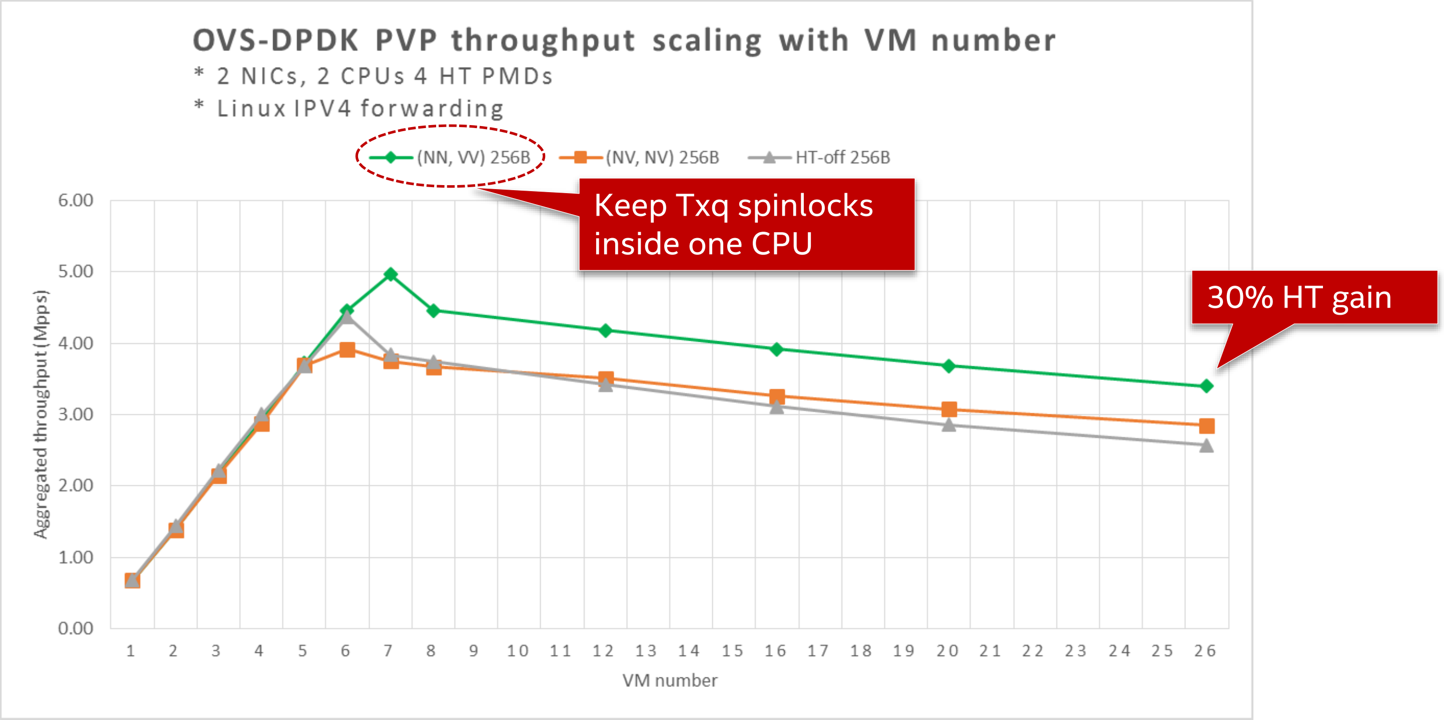
But don't spoil it!

- Keep spinlocks inside one CPU

Side effects: Again, increase of latency



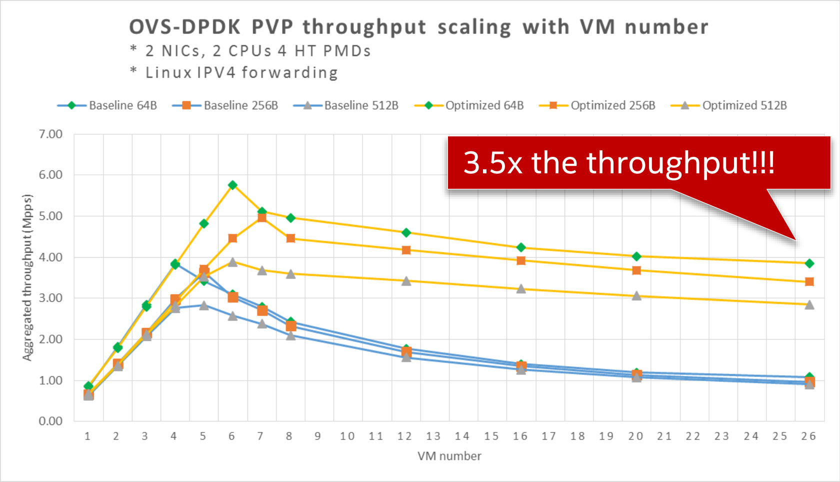
Hyper-Threading Gain

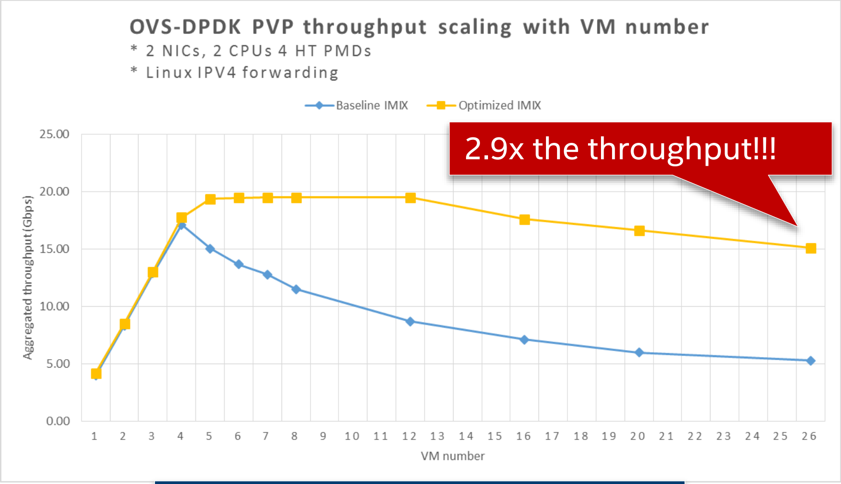


* Test and System Configurations: Estimates are based on internal Intel analysis using Intel® Server Board S2600WT, Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, Intel® 82599ES 10 Gigabit Ethernet Controller

Finally, With All Due Optimization

About 3x the throughput





IMIX: Uniform distribution random [64, 1518]

* Test and System Configurations: Estimates are based on internal Intel analysis using Intel® Server Board S2600WT, Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz, Intel® 82599ES 10 Gigabit Ethernet Controller

Latency Is The Price To Pay

Latency is increased due to

- Extra overhead of buffering mechanism
- Lower frequency due to Hyper-Threading

Latency is decided by

- Guests per PMD core (Round-Robin)
- Traffic pattern (How fragmented?)
- Tx-Buffering timeout settings

No silver bullet, only tradeoffs and priorities

Deployment Guideline

Enable Vhost Tx-Buffering for throughput

- Enable for throughput peaks dynamically
- Evaluate timeout settings carefully

Allocate CPU based on expected traffic model

Utilize Hyper-Threading for PMDs

- For 1 CPU, 1 HT for NIC, 1 HT for Vhost
- For 2 CPUs, 2 HTs on 1 CPU for NIC, 2 HTs on another for Vhost
- *For more than 2 CPUs, per case tuning is required*

experience
what's inside™