200GbE 時代に再考する ハイパーバイザネットワーク設計

Akira KAMIO

whoami

神尾　皓 (Akira KAMIO)

- 株式会社サイバーエージェント
  - CyberAgent group Infrastructure Unit (CIU)
  - Cycloud Div, Compute Team
- 2020/06 中途入社
  - 入社以来、一貫してIaaS基盤の開発/運用業務に従事
  - QEMU/KVM や高集約コンピュートノードの最適化に興味がある

Contents

CyberAgent

本発表の結論

- 新 HV では 100GbE x2 の物理帯域を前提に、HV あたり 170Gbps 級の実運用帯域が必要
- vDPA / SR-IOV も検討したが、互換性・運用要件から　OVS-DPDK を採用
- EPYC 9655P + ConnectX-6 Dx + OVS-DPDK により 3c6t で 170Gbps, 4c8t で 200Gbps を確認
- 実運用では VM 収容効率とのバランスから　3c6t / 170Gbps 構成を採用

1

プライベートクラウドに求められるものの変化

CyberAgent

これまでのプライベートクラウド (1)

- サイバーエージェントではサービス開発者がインフラを自由に選べる
  - プライベートクラウドは　AWS / Google Cloud / Azure と比較される

プライベートクラウドが選ばれるには明確なメリットが必要

これまでのプライベートクラウド (2)

- 機能性
  - パブリッククラウド ＞ プライベートクラウド
- 性能
  - パブリッククラウド ＞ プライベートクラウド
- 運用性
  - パブリッククラウド ＞ プライベートクラウド
- コスト
  - パブリッククラウド ＜ プライベートクラウド

サービス開発者がコスト以外でプライベートクラウドを選ぶ合理性は低い

プライベートクラウドの式年遷宮

- 約7年ぶりとなるプライベートクラウドの刷新
- 目標は「パブリッククラウドと戦えるプライベートクラウド」
  - 従来の主な優位性はコストであったが、機能面・性能面を強化

- IaaS 基盤においては特に **性能面を重点的に刷新**

既存 IaaS の性能と AWS

旧プライベートクラウド (EPYC Rome)

Flavor	Core (thread)	Network Gbps
a1.large	2	1.2
a1.xlarge	4	2.4
a1.2xlarge	8	4.8
a1.3xlarge	12	7.2
a1.4xlarge	16	9.6

AWS 最新世代 (EPYC Turin)

Flavor	Core (thread)	Network Gbps
m8a.medium	1	0.52
m8a.large	2	0.937
m8a.xlarge	4	1.875
m8a.2xlarge	8	3.75
m8a.4xlarge	16	7.5
m8a.8xlarge	32	15
m8a.12xlarge	48	22.5
m8a.16xlarge	64	30
m8a.24xlarge	96	40
m8a.32xlarge	128	50

同一コア数では同等程度のスペックに見えるが
現実は旧プライベートクラウドでは ホスト全体で 20Gbps 程度で頭打ち

既存 IaaS の性能と AWS

旧プライベートクラウド (EPYC Rome)

Flavor	Core (thread)	Network Gbps
a1.large	2	1.2
a1.xlarge	4	2.4
a1.2xlarge	8	4.8
a1.3xlarge	12	7.2
a1.4xlarge	16	9.6

AWS 最新世代 (EPYC Turin)

Flavor	Core (thread)	Network Gbps
m8a.medium	1	0.52
m8a.large	2	0.937
m8a.xlarge	4	1.875
m8a.2xlarge	8	3.75
m8a.4xlarge	16	7.5
		15
		22.5
		30
		40
		50

これらの VM サービストラフィックに加えて
ストレージ・マイグレーショントラフィックが加わる

=> 高帯域なハイパーバイザネットワークが必要

現実は旧プライベートクラウドでは ホスト全体で 20Gbps 程度で頭打ち

2

ハイパーバイザの選定と性能

CyberAgent

ハイパーバイザ選定

- CPU
  - なるべくパフォーマンスが高いこと
  - なるべくコアが多いこと
- Memory: 1thread 4GiB or 8GiB
- Storage:　NVMe RAID
- Network: 100GbE x2 NIC

ハイパーバイザ選定

- CPU
  - なるべくパフォーマンスが高いこと
  - なるべくコアが多いこと
- Memory: 1thread 4GiB or 8GiB
- Storage:　NVMe RAID
- Network: 100GbE x2 NIC
- NUMA:　1NUMA に収まること

ハイパーバイザ選定

- CPU
  - なるべくパフォーマンスが高いこと
  - なるべくコアが多いこと
- Memory: 1thread 4GiB or 8GiB
- Storage:　NVMe RAID
- Network: 100GbE x2 NIC
- NUMA:　1NUMA に収まること

ハイパーバイザ選定

- CPU
  - なるべくパフォーマンスが高いこと
  - なるべくコアが多いこと
- Memory: 1thread 4GiB or 8GiB
- Storage:　NVMe RAID
- Network: 100GbE x2 NIC
- NUMA:　1NUMA に収まること -> 1 socket

1 socket CPU の候補

- EPYC 9654P
  - Genoa, Zen4, 96cores 192threads, 360W
- EPYC 9754
  - Bergamo, Zen4c, 128cores 256threads, 360W
- EPYC 9655P
  - Turin, Zen5, 96cores 192threads, 360W

1 socket CPU の候補

- EPYC 9654P
  - Genoa, Zen4, 96cores 192threads, 360W
- EPYC 9754
  - Bergamo, Zen4c, 128cores 256threads, 360W
- EPYC 9655P 👑
  - Turin, Zen5, 96cores 192threads, 360W

圧倒的な CPU パフォーマンス
1.5x vs. Genoa, Bergamo

ハイパーバイザ選定

- CPU
  - なるべくパフォーマンスが高いこと
  - なるべくコアが多いこと
- Memory: 1thread 4GiB or 8GiB
- Storage:　NVMe RAID
- Network: 100GbE x2 NIC
- NUMA:　1NUMA に収まること -> 1 socket

NIC の要件

- **100GbE x2 port**
- **複数の仮想ネットワーク方式を検証できること**
  - **vDPA / SR-IOV / OVS-DPDK**
- **HW offload 経路を持つこと**
  - **OVS dataplane offload / switchdev**
  - **virtio device offload**

NIC の候補

- **NVIDIA ConnectX-6 Dx**

  100GbE x2 対応 / DPDK / SR-IOV / switchdev / vDPA への展開余地

  OpenStack / OVS-DPDK 構成での利用実績が多い

- **Intel E810**

  100GbE x2 対応 / Linux / DPDK 実績あり

  一方で vDPA / OVS HW offload まで含めた構成検証は要確認

- **NVIDIA BlueField DPU**

  NIC 上で OVS / virtio を動かせる可能性

  ただしコスト・運用複雑性・ベンダーロックインが大きい

NIC の候補

- **NVIDIA ConnectX-6 Dx**
  - 100GbE x2 対応 / DPDK / SR-IOV / switchdev / vDPA への展開余地
  - OpenStack / OVS-DPDK 構成での利用実績が多い
- **Intel E810**
  - 100GbE x2 対応 / Linux / DPDK 実績あり
  - 一方で vDPA / OVS HW offload まで含めた構成検証は要確認
- **NVIDIA BlueField DPU**
  - NIC 上で OVS / virtio を動かせる可能性
  - ただしコスト・運用複雑性・ベンダーロックインが大きい

NIC の候補

- **NVIDIA ConnectX-6 Dx**

  100GbE x2 対応 / DPDK / SR-IOV / switchdev / vDPA への展開余地

  OpenStack / OVS-DPDK 構成での利用実績が多い

- **Intel E810**

  100GbE x2 対応 / Linux / DPDK 実績あり

  一方で vDPA / OVS HW offload まで含めた構成検証は要確認

- **NVIDI**

  NIC 上で

  ただしコスト・運用複雑性・ベンダーロックインが大きい

帯域・機能・運用実績・将来性のバランスから
NVIDIA ConnectX-6 Dx を選定

最終的なハイパーバイザ物理構成

CPU: EPYC 9655P (Zen5, 96cores 192threads)

MEM: 768GiB (1thread 4GiB)

Storage: 3.2TiB NVMe RAID1

NIC: NVIDIA ConnectX-6 Dx 100GbE 2port

最終的なハイパーバイザ物理構成

CPU: EPYC 9655P (Zen5, 96cores 192threads)

MEM: 768GiB (1thread 4GiB)

Storage: 3.2TiB NVMe RAID1

NIC: NVIDIA ConnectX-6 Dx 100GbE 2port

物理構成は決まったので 200Gbps 級の帯域を活かせる
ハイパーバイザ面を設計していく

（再掲）既存 IaaS の性能と AWS

旧プライベートクラウド
(EPYC Rome)

Flavor	Core (thread)	Network Gbps
a1.large	2	1.2
a1.xlarge	4	2.4
a1.2xlarge	8	4.8
a1.3xlarge	12	7.2
a1.4xlarge	16	9.6

AWS 最新世代
(EPYC Turin)

Flavor	Core (thread)	Network Gbps
m8a.medium	1	0.52
m8a.large	2	0.937
m8a.xlarge	4	1.875
m8a.2xlarge	8	3.75
m8a.4xlarge	16	7.5
m8a.8xlarge	32	15
m8a.12xlarge	48	22.5
m8a.16xlarge	64	30
m8a.24xlarge	96	40
m8a.32xlarge	128	50

同一コア数では同等程度のスペックに見えるが
現実は旧プライベートクラウドでは ホスト全体で 20Gbps 程度で頭打ち

（再掲）既存 IaaS の性能と AWS

旧プライベートクラウド (EPYC Rome)

Flavor	Core (thread)	Network Gbps
a1.large	2	1.2
a1.xlarge	4	2.4
a1.2xlarge	8	4.8
a1.3xlarge	12	7.2
a1.4xlarge	16	9.6

AWS 最新世代 (EPYC Turin)

Flavor	Core (thread)	Network Gbps
m8a.medium	1	0.52
m8a.large	2	0.937
m8a.xlarge	4	1.875
m8a.2xlarge	8	3.75
m8a.4xlarge	16	7.5
m8a.8xlarge	32	15
m8a.12xlarge	48	22.5
m8a.16xlarge	64	30
m8a.24xlarge	96	40
m8a.32xlarge	128	50

EPYC 9655P (96c192t) では VM トラフィック 40Gbps x2 = 80Gbps がターゲット

（再掲）既存 IaaS の性能と AWS

旧プライベートクラウド (EPYC Rome)

Flavor	Core (thread)	Network Gbps
a1.large	2	1.2
a1.xlarge	4	2.4
a1.2xlarge	8	4.8
a1.3xlarge	12	7.2
a1.4xlarge	16	9.6

ストレージ：　MAX 50Gbps
マイグレーション：　MAX 40Gbps

AWS 最新世代 (EPYC Turin)

Flavor	Core (thread)	Network Gbps
m8a.medium	1	0.52
m8a.large	2	0.937
m8a.xlarge	4	1.875
m8a.2xlarge	8	3.75
m8a.4xlarge	16	7.5
m8a.8xlarge	32	15
m8a.12xlarge	48	22.5
m8a.16xlarge	64	30
m8a.24xlarge	96	40
m8a.32xlarge	128	50

EPYC 9655P (96c192t) では VM トラフィック 40Gbps x2 = 80Gbps がターゲット

（再掲）既存 IaaS の性能と AWS

旧プライベートクラウド
(EPYC Rome)

Flavor	Core (thread)	Network Gbps
a1.large	2	1.2
a1.xlarge		
a1.2xlarge		
a1.3xlarge		
a1.4xlarge		

AWS 最新世代
(EPYC Turin)

Flavor	Core (thread)	Network Gbps
m8a.medium	1	0.52
		0.937
		1.875
		3.75
		7.5
m8a.8xlarge	32	15
m8a.12xlarge	48	22.5
m8a.16xlarge	64	30
m8a.24xlarge	96	40
m8a.32xlarge	128	50

80Gbps + 50Gbps + 40Gbps = 170Gbps ！

ストレージ：　　MAX 50Gbps
マイグレーション：　MAX 40Gbps

EPYC 9655P (96c192t) では VM トラフィック 40Gbps x2 = 80Gbps がターゲット

（再掲）既存 IaaS の性能と AWS

旧プライベートクラウド
(EPYC Rome)

Flavor	Core (thread)	Network Gbps
a1.large	2	1.2
a1.xlarge		
a1.2xlarge		
a1.3xlarge		
a1.4xlarge		

AWS 最新世代
(EPYC Turin)

Flavor	Core (thread)	Network Gbps
m8a.medium	1	0.52
		0.937
		1.875
		3.75
		7.5
m8a.8xlarge	32	15
m8a.12xlarge	48	22.5
m8a.16xlarge	64	30
m8a.24xlarge	96	40

80Gbps + 50Gbps + 40Gbps = 170Gbps ！

ストレージ :　　MAX 50Gbps
マイグレーション :　MAX 40Gbps

=> ハイパーバイザあたり　170Gbps の帯域が必要

3

仮想ネットワーク方式の選定

CyberAgent

仮想ネットワーク方式候補

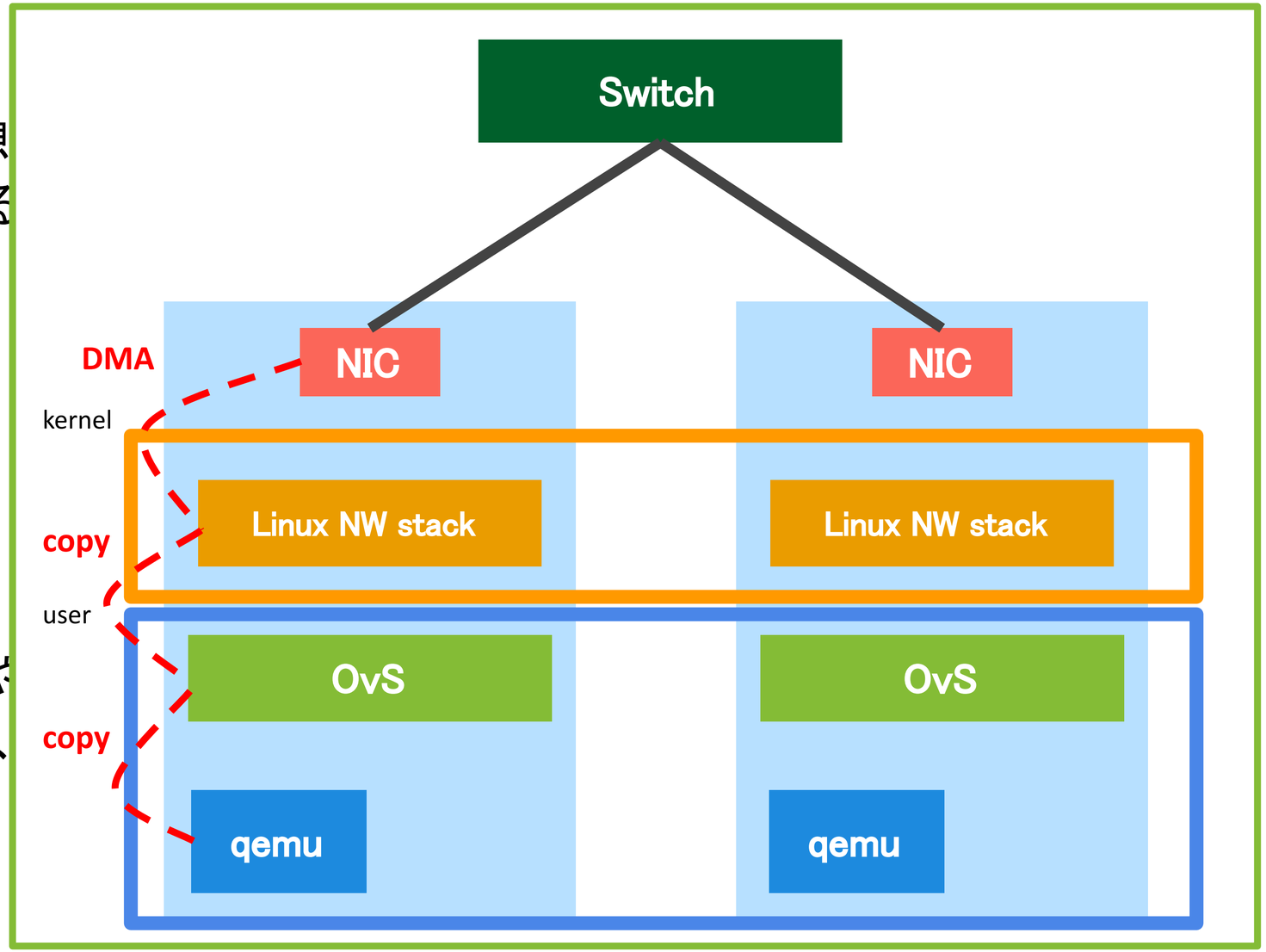
- OpenvSwitch
- OpenvSwitch + DPDK + vhost-user
- OpenvSwitch + SR-IOV
- OpenvSwitch + vDPA Kernel Framework

OpenvSwitch

- **特徴**
  - オープンソースの OpenFlow 実装の仮想スイッチ
  - linuxbridge と異なり user 空間で動作する
- **性能**
  - ～20Gbps
- **Pros.**
  - 手軽に構築できる
- **Cons.**
  - メモリコピーが多く発生する
    - softirq が多い -> CPU 使用率が上がりやすい
    - user 空間で実行されるプロセスのため、タスクスケジューリングの影響を受けやすい
    - **レイテンシが安定しない**

OpenvSwitch

- **特徴**
  - オープンソースの OpenFlow 実装の仮想
  - linuxbridge と異なり user 空間で動作する
- **性能**
  - ～20Gbps
- **Pros.**
  - 手軽に構築できる
- **Cons.**
  - メモリコピーが多く発生する
    - softirq が多い -> CPU 使用率が上がりや
    - user 空間で実行されるプロセスのため、
    - **レイテンシが安定しない**



OpenvSwitch + DPDK + vhost-user

- **特徴**
  - OpenvSwitch と DPDK (Data Plane Development Kit) を組み合わせたもの
  - NFV で用いられることも多い
  - ポーリングプロセスを user 空間で動作させ、パケットを処理
  - vhost-user を用いることで shared memory を介してパケットをやり取りできる
- **性能**
  - ～50Gbps
- **Pros.**
  - OpenvSwitch をそのまま使うよりも圧倒的に**低レイテンシ**
  - OpenvSwitch の機能をそのまま利用できる
- **Cons.**
  - ポーリングプロセスのために CPU コアを割り当てる必要がある（余分にリソースが必要）

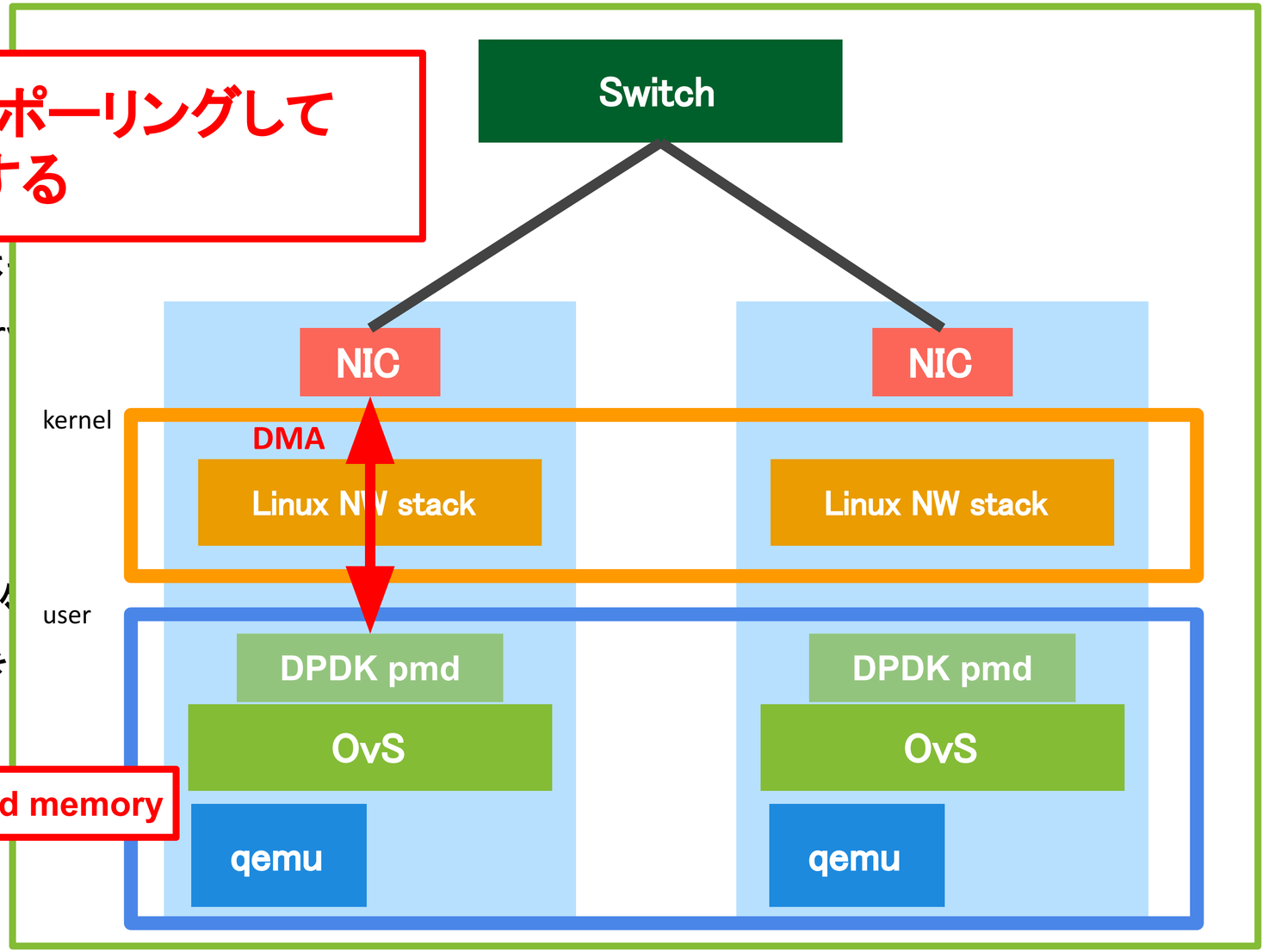
OpenvSwitch + DPDK + vhost-user

user 空間にあるプロセスで NIC をポーリングして kernel 空間をバイパスする

- ポーリングプロセスを user 空間で動作さ
- vhost-user を用いることで shared memor

- **性能**
  - ～50Gbps
- **Pros.**
  - OpenvSwitch をそのまま使うよりも圧倒的
  - OpenvSwitch の機能をそのまま利用でき
- **Cons.**
  - ポーリングプロセスの

OvS - vNIC 間は shared memory

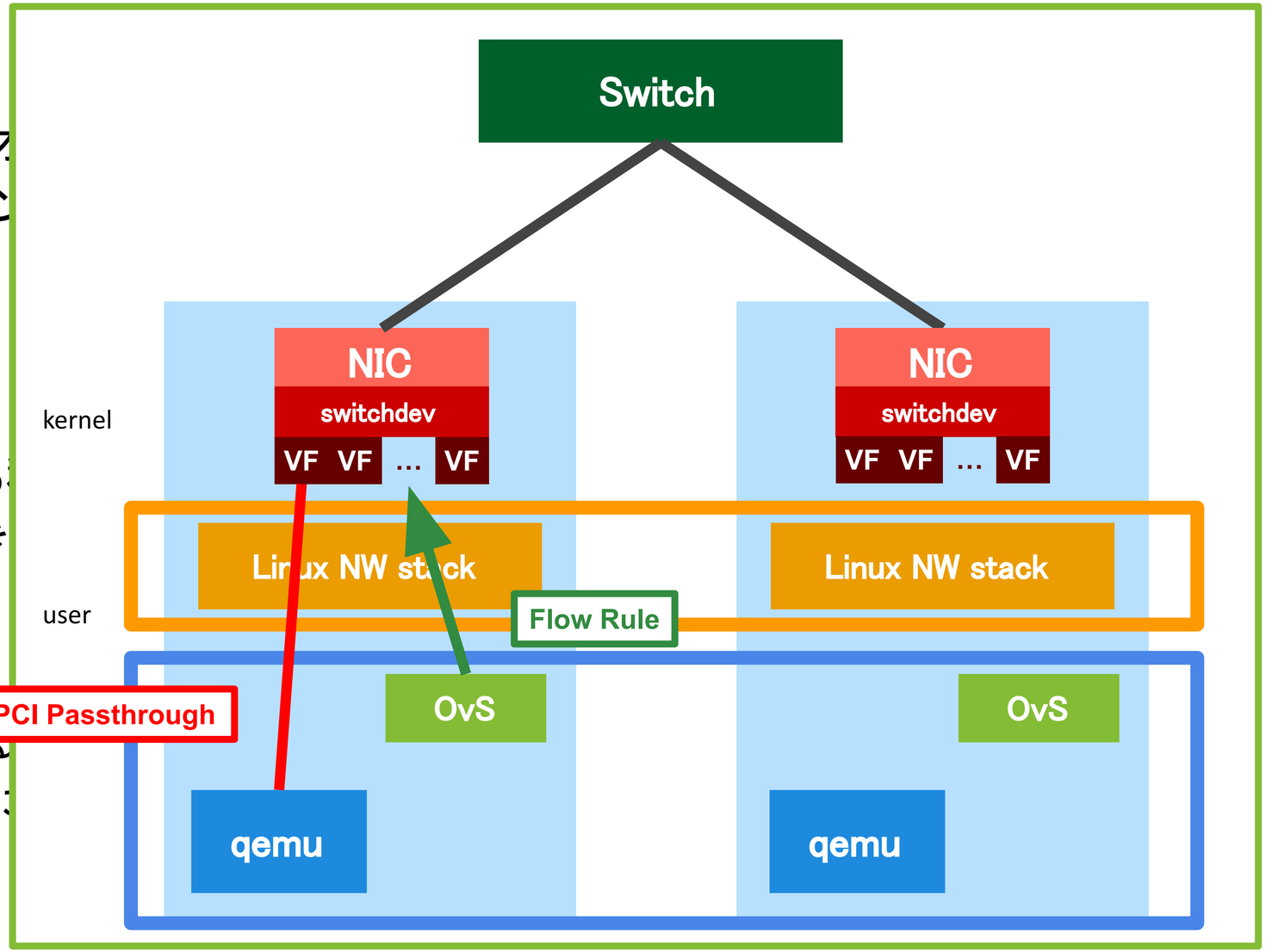


OpenvSwitch + SR-IOV

- **特徴**
  - OpenvSwitch を NIC 等のハードウェアでオフロードする
  - OpenvSwitch のプロセスは OpenFlow コントローラとしての役割にほぼ専念する
- **性能**
  - ～200Gbps (wire-rate)
- **Pros.**
  - 100Gbps x2 でワイヤーレートを目指せる程度に高速
  - OpenvSwitch の機能をそのまま利用できる
- **Cons.**
  - ライブマイグレーションができない
  - 使い方によってはベンダーロックインする
  - VF (SR-IOV で分割したデバイス) の利用にドライバが必要（古い OS だと疎通しない）

OpenvSwitch + SR-IOV

- **特徴**
  - OpenvSwitch を NIC 等のハードウェアで
  - OpenvSwitch のプロセスは OpenFlow コン
- **性能**
  - ～200Gbps (wire-rate)
- **Pros.**
  - 100Gbps x2 でワイヤーレートを目指せる
  - OpenvSwitch の機能をそのまま利用でき
- **Cons.**
  - ライブマイグレーションができない
  - 使い方によってはベンダーロックインする
  - VF (SR-IOV で分割したデバイス) の利用に



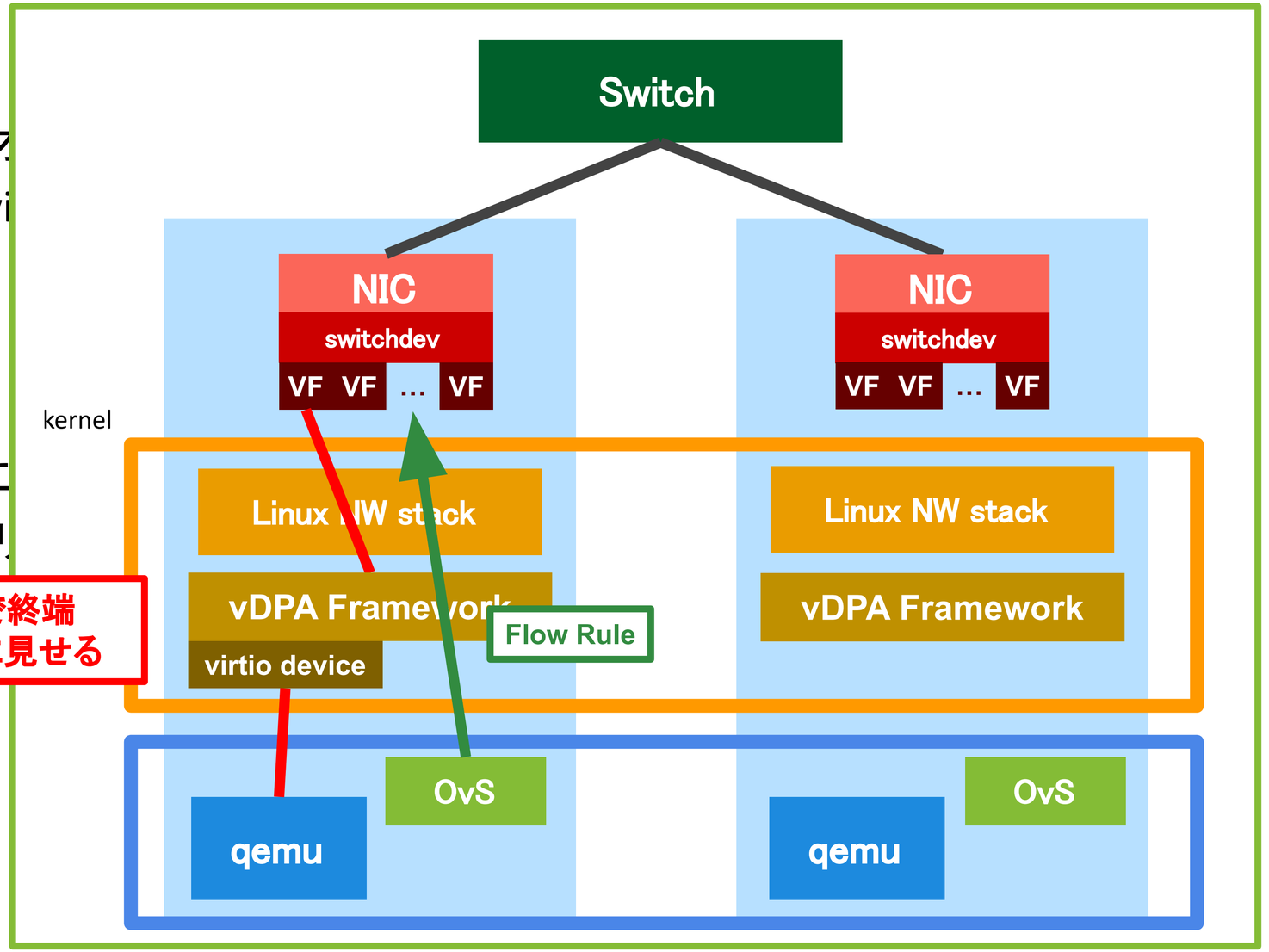
OpenvSwitch + vDPA Kernel Framework

- **特徴**
  - OpenvSwitch を NIC 等のハードウェアでオフロードする
  - VM へ VF を渡さず host kernel で終端、virtio デバイスを渡す
- **性能**
  - ～200Gbps？
- **Pros.**
  - OpenvSwitch HW Offload をより汎用的に利用できる
  - virtio driver さえあれば HW Offload のメリットを享受できる
- **Cons.**
  - ドキュメントが揃っていない

OpenvSwitch + vDPA Kernel Framework

- **特徴**
  - OpenvSwitch を NIC 等のハードウェアで
  - VM へ VF を渡さず host kernel で終端、vi
- **性能**
  - ～200Gbps？
- **Pros.**
  - OpenvSwitch HW Offload をより汎用的に
  - virtio driver さえあれば HW Offload のメ
- **Cons.**
  - ドキュメントが揃っていない

VF を vDPA Framework で終端
virtio device として qemu に見せる



構成まとめ

	OpenvSwitch	OvS + DPDK	OvS + SR-IOV	OvS + vDPA Framework
Latency	High	Low	Low	Low
Speed	20Gbps	50Gbps	200Gbps (wire-rate)	200Gbps (wire-rate)
Live Migration	Yes	Yes	No	Yes
virtio support	Yes	Yes	No	Yes
vendor lock	No	No	Yes	No
Manageability	Mid	Mid	Low	?
CPU usage	Mid	High (using dedicated core)	Low	Low

構成まとめ

	OpenvSwitch	OvS + DPDK	OvS + SR-IOV	OvS + vDPA Framework
Latency	High	Low	Low	Low
Speed	20Gbps	50Gbps	200Gbps (wire-rate)	200Gbps (wire-rate)
Live Migration	Yes	Yes	No	Yes
virtio support	Yes	Yes	No	Yes
vendor lock	No	No	Yes	No
Manageability	Mid	Mid	Low	?
CPU usage	Mid	High (using dedicated core)	Low	Low

OpenvSwitch + vDPA Kernel Framework が良さそう…！

仮想ネットワーク方式候補

- OpenvSwitch
- OpenvSwitch + DPDK + vhost-user
- OpenvSwitch + SR-IOV
- OpenvSwitch + vDPA Kernel Framework

仮想ネットワーク方式候補

- ~~OpenvSwitch~~ <- プロダクション向け構成ではないため除外
- OpenvSwitch + DPDK + vhost-user
- OpenvSwitch + SR-IOV
- OpenvSwitch + vDPA Kernel Framework

OpenvSwitch + DPDK + vhost-user

- チューニング次第で高帯域を狙える
- CPU コアを余分に使ってしまうのは嬉しくない　…

OpenvSwitch + SR-IOV

- とても高速なのはわかっているので採用したい　…！

- ゲスト OS の互換性を考えると採用できない
  - すべてのゲストが　Mellanox NIC VF と喋れるとは限らない
- ライブマイグレーションができない

OpenvSwitch + vDPA Kernel Framework

- とても高速ないい感じ　NW が構成できそう …！

- 実際に構築しようとしたところ　…
  - NVIDIA NIC のドライバ周りで
    MLNX_OFED -> DOCA への過渡期で **構築不可**
  - mlx5_vdpa driver が dummy driver になっていたり …

構築時点のドライバ／運用要件では採用困難

仮想ネットワーク方式候補

- ~~OpenvSwitch~~
- OpenvSwitch + DPDK + vhost-user
- OpenvSwitch + SR-IOV
- OpenvSwitch + vDPA Kernel Framework

仮想ネットワーク方式候補

- ~~**OpenvSwitch**~~
- **OpenvSwitch + DPDK + vhost-user -> CPU コア使う**
- **OpenvSwitch + SR-IOV -> 互換性なし**
- **OpenvSwitch + vDPA Kernel Framework -> 構築不可**

仮想ネットワーク方式候補

- ~~OpenvSwitch~~
- OpenvSwitch + DPDK + vhost-user -> CPU コア使う
- OpenvSwitch + SR-IOV -> 互換性なし
- OpenvSwitch + vDPA Kernel Framework -> 構築不可

ほな、消去法で OpenvSwitch + DPDK かな…

仮想ネットワーク方式候補

- ~~OpenvSwitch~~
- OpenvSwitch + DPDK + vhost-user -> CPU コア使う
- OpenvSwitch + SR-IOV -> 互換性なし
- Op　　可

OVS-DPDK 構成で 200Gbps を目指すことに

ほな、消去法で OpenvSwitch + DPDK かな…

4

OVS-DPDK データプレーン設計

（再掲）OpenvSwitch + DPDK + vhost-user

- **特徴**
  - OpenvSwitch と DPDK (Data Plane Development Kit) を組み合わせたもの
  - NFV で用いられることも多い
  - ポーリングプロセスを user 空間で動作させ、パケットを処理
  - vhost-user を用いることで shared memory を介してパケットをやり取りできる
- **性能**
  - ～50Gbps
- **Pros.**
  - OpenvSwitch をそのまま使うよりも圧倒的に**低レイテンシ**
  - OpenvSwitch の機能をそのまま利用できる
- **Cons.**
  - ポーリングプロセスのために CPU コアを割り当てる必要がある（余分にリソースが必要）

（再掲）OpenvSwitch + DPDK + vhost-user

- **特徴**
  - OpenvSwitch と DPDK (Data Plane Development Kit) を組み合わせたもの
  - NFV で用いられることも多い
  - ポーリングプロセスを user 空間で動作させ、パケットを処理
  - vhost-user を用いることで shared memory を介してパケットをやり取りできる
- **性能**
  - ～50Gbps
- **Pros.**
  - OpenvSwitch をそのまま使うよりも圧倒的に**低レイテンシ**
  - OpenvSwitch の機能をそのまま利用できる
- **Cons.**
  - ポーリングプロセスのために CPU コアを割り当てる必要がある（余分にリソースが必要）

DPDK のパフォーマンス特性

- DPDK は CPU コアを割り当てるほどパフォーマンスが上がる
  - ただし NIC の queue 数や NUMA 配置などにも依存する

- 性能を引き出すには CPU コア配置が重要
  - 同一 NUMA node、同一 L3 cache を共有するコアを利用
  - 配置が不適切だと、メモリアクセスやキャッシュ効率が悪化し性能が頭打ちになる

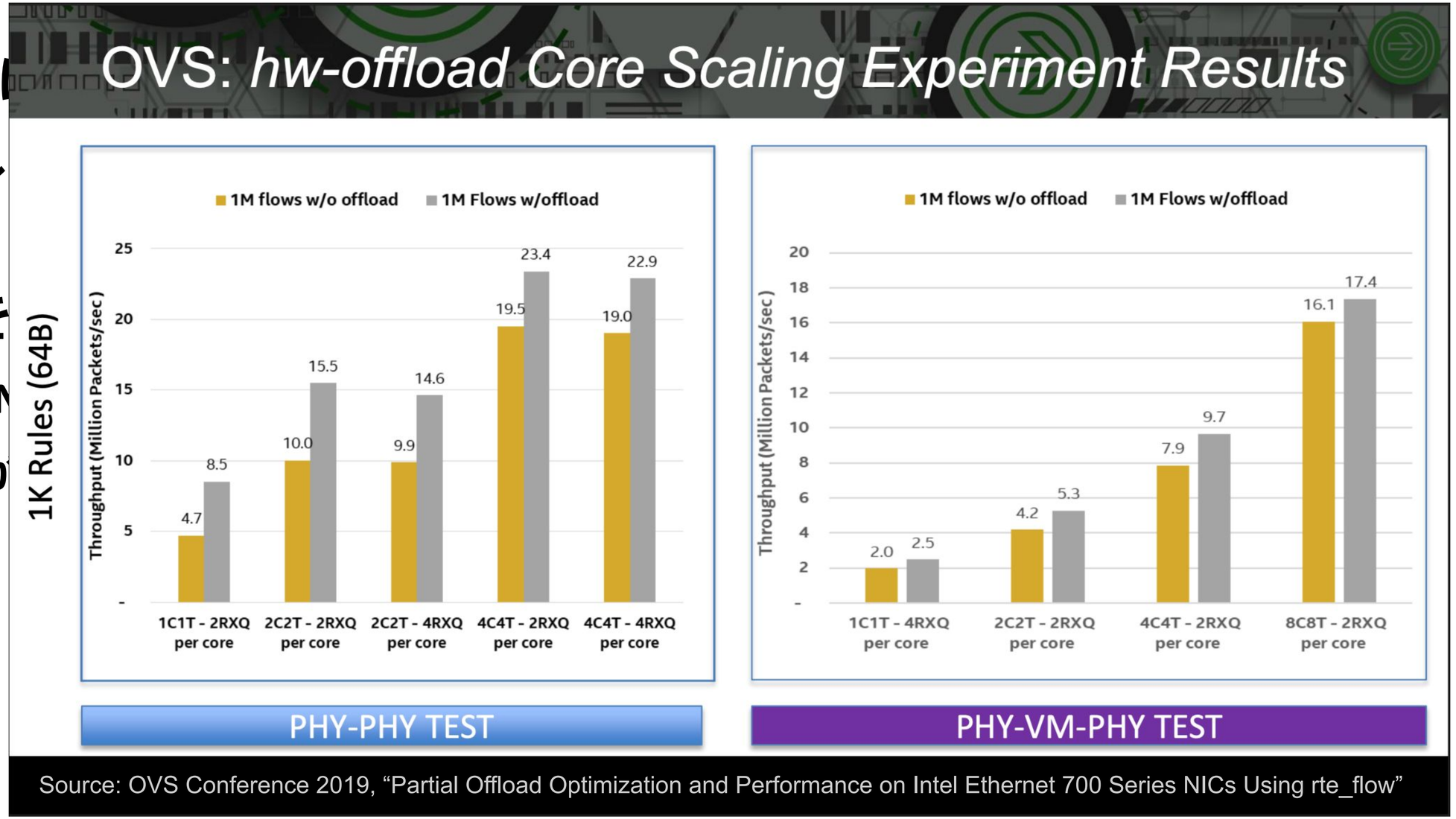
DPDK のパフォーマンス特性

- DPDK は CPU コアを割り当てるほどパフォーマンスが上がる
  - ただし NIC の queue 数や NUMA 配置などにも依存する

- 性能を引き出すには CPU コア配置が重要
  - 同一 NUMA node、同一 L3 cache を共有するコアを利用
  - 配置が不適切だと、メモリアクセスやキャッシュ効率が悪化し性能が頭打ちになる

DPDK のパフォーマンス特性

- DPDK
  - ただし
- 性能を
  - 同一 N
  - 配置が



DPDK のパフォーマンス特性

- DPDK は CPU コアを割り当てるほどパフォーマンスが上がる
  - ただし NIC の queue 数や NUMA 配置などにも依存する

- 性能を引き出すには CPU コア配置が重要
  - 同一 NUMA node、同一 L3 cache を共有するコアを利用
  - 配置が不適切だと、メモリアクセスやキャッシュ効率が悪化し性能が頭打ちになる

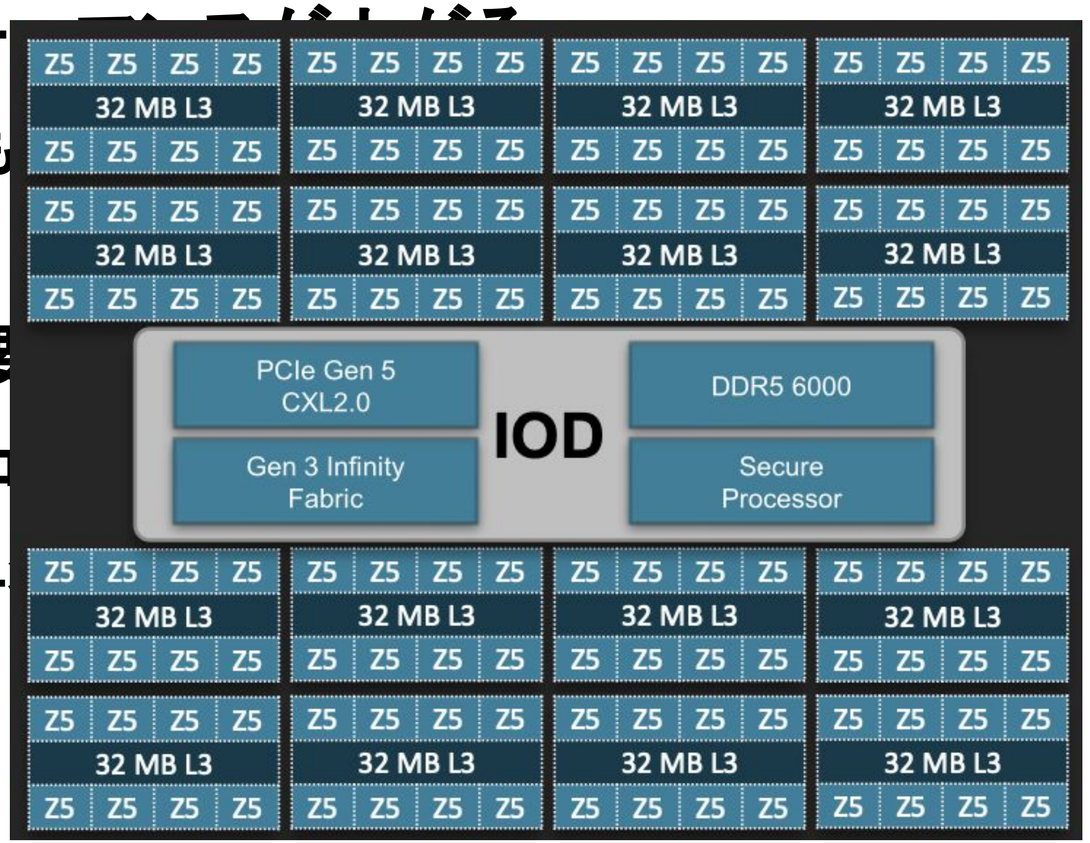
DPDK のパフォーマンス特性

- DPDK は CPU コアを割り当てるほどパフォーマンスが上がる
  - ただし NIC の queue 数や NUMA 配置などにも依存する

- 性能を引き出すには CPU コア配置が重要
  - 同一 NUMA node、同一 L3 cache を共有するコアを利用
  - 配置が不適切だと、メモリアクセスやキャッシュ効率が悪化し性能が頭打ちになる

DPDK のパフォーマンス特性

- DPDK は CPU コアを割り当てるほどパフォ
  - ただし NIC の queue 数や NUMA 配置などにも
- 性能を引き出すには CPU コア配置が重要
  - 同一 NUMA node、同一 L3 cache を共有するコ
  - 配置が不適切だと、メモリアクセスやキャッシュ

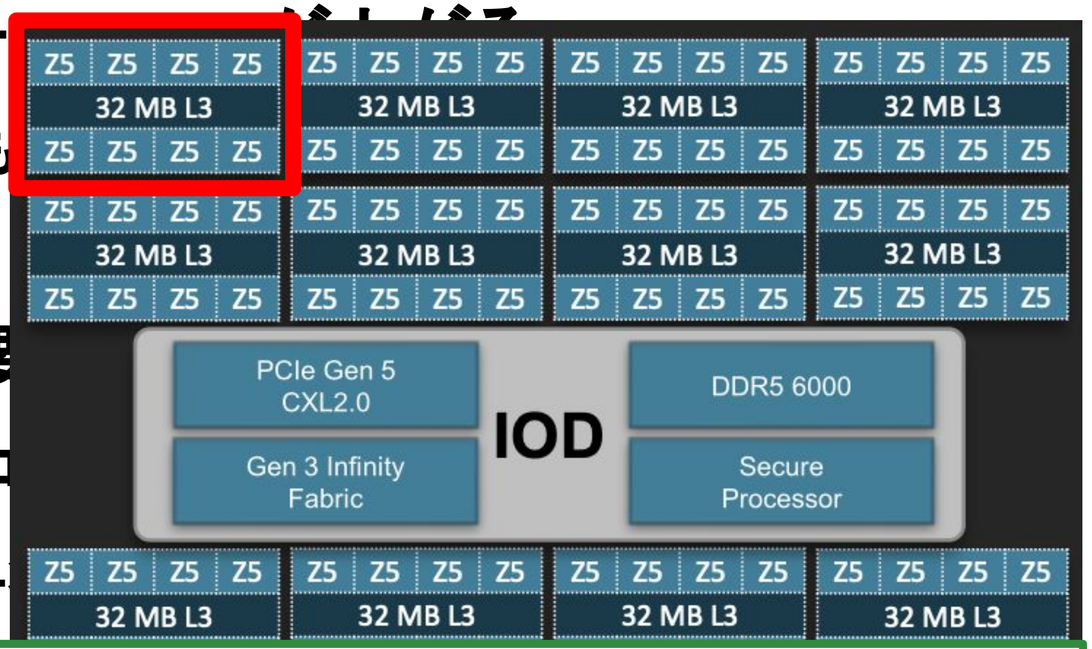


DPDK のパフォーマンス特性

- DPDK は CPU コアを割り当てるほどパフォ
  - ただし NIC の queue 数や NUMA 配置などにも
- 性能を引き出すには CPU コア配置が重要
  - 同一 NUMA node、同一 L3 cache を共有するコ
  - 配置が不適切だと、メモリアクセスやキャッシュ

DPDK のパフォーマンス特性

- DPDK は CPU コアを割り当てるほどパフォ
  - ただし NIC の queue 数や NUMA 配置などにも
- 性能を引き出すには CPU コア配置が重要
  - 同一 NUMA node、同一 L3 cache を共有するコ
  - 配置が不適切だと、メモリアクセスやキャッシュ



EPYC Turin は L3 cache を 8cores 16threads で共有
ここから PMD CPU コアをアサインすれば良い

PMD アサインする CPU のバランス

- PMD に CPU コアを割り当てるほど、仮想スイッチ性能は向上する
  - 一方で、PMD 用に確保した CPU コアは VM に割り当てられない

- そのため、PMD コア数は性能と収容効率のトレードオフになる
  - 少ない PMD コア数で 200Gbps 級の帯域を実現したい

PMD アサインする CPU のバランス

- PMD に CPU コアを割り当てるほど、仮想スイッチ性能は向上する
  - 一方で、PMD 用に確保した CPU コアは VM に割り当てられない

- そのため、PMD コア数は性能と収容効率のトレードオフになる
  - 少ない PMD コア数で 200Gbps 級の帯域を実現したい

2cores 4threads で 200Gbps の実現を目標に

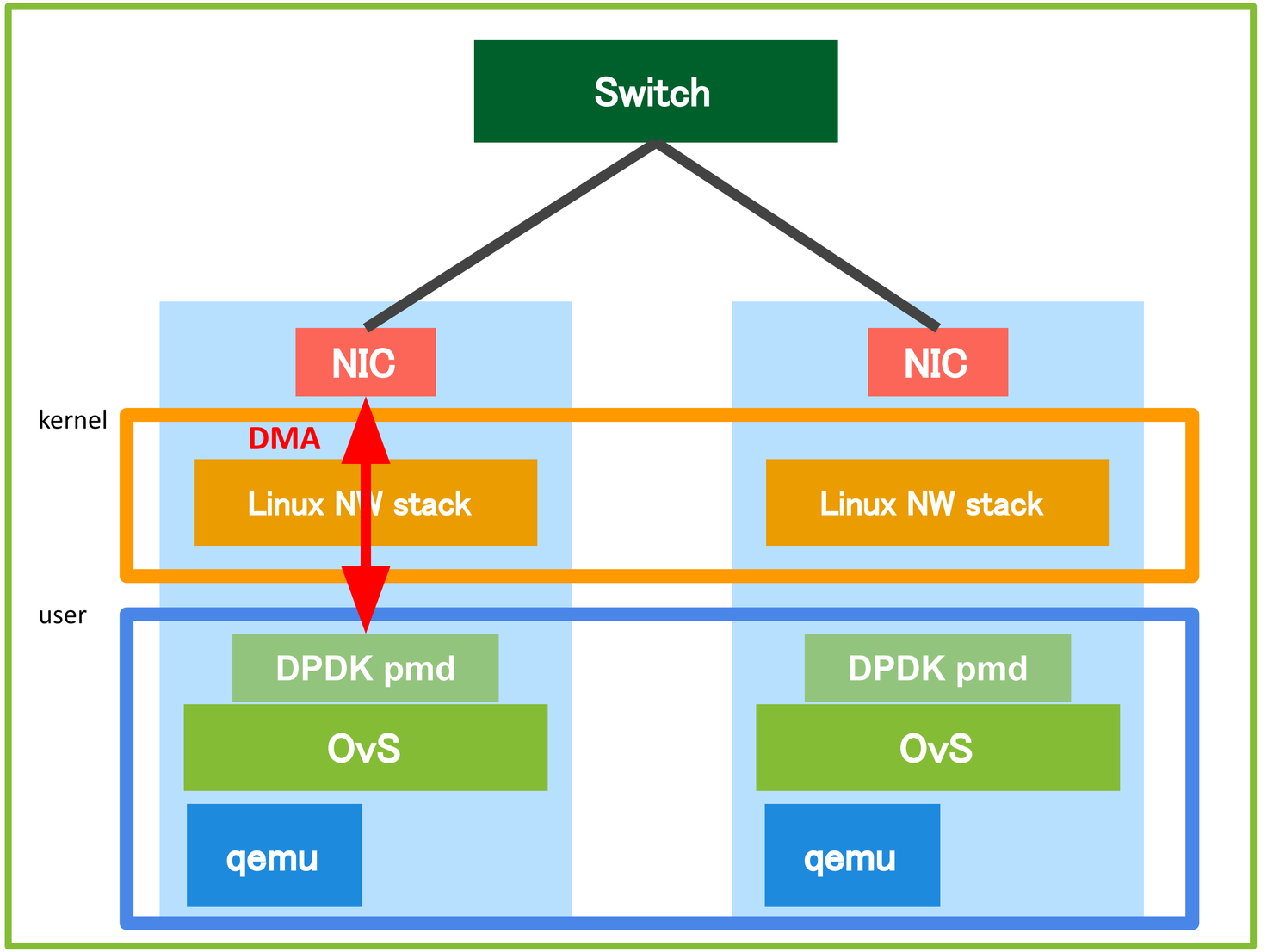
5

実現したハイパーバイザ構成

CyberAgent

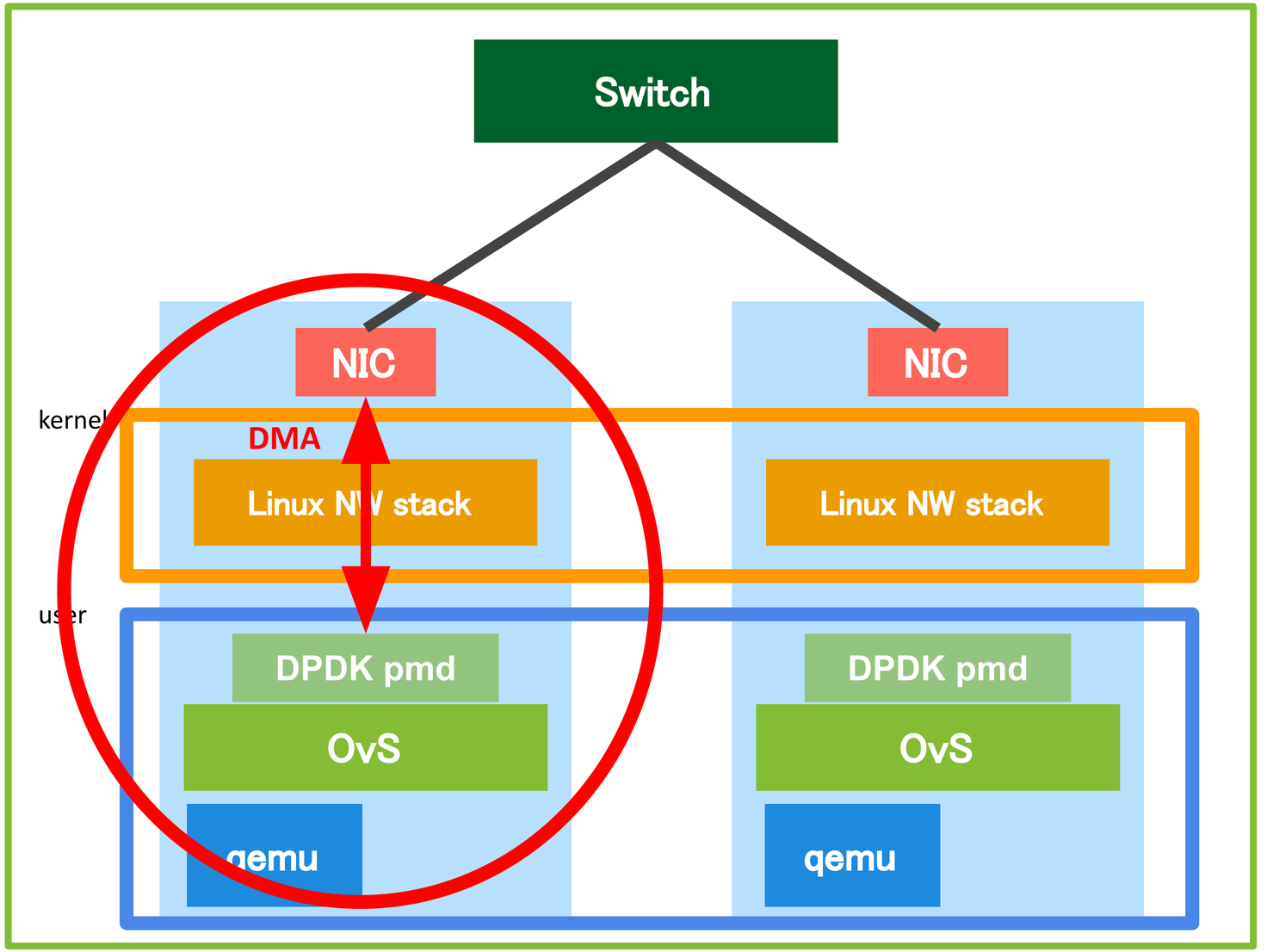
実現した OpenvSwitch 構成 (1)

- PMD 3cores 6threads
  - dpdkvhostuserclient
    - qemu をホストとして接続

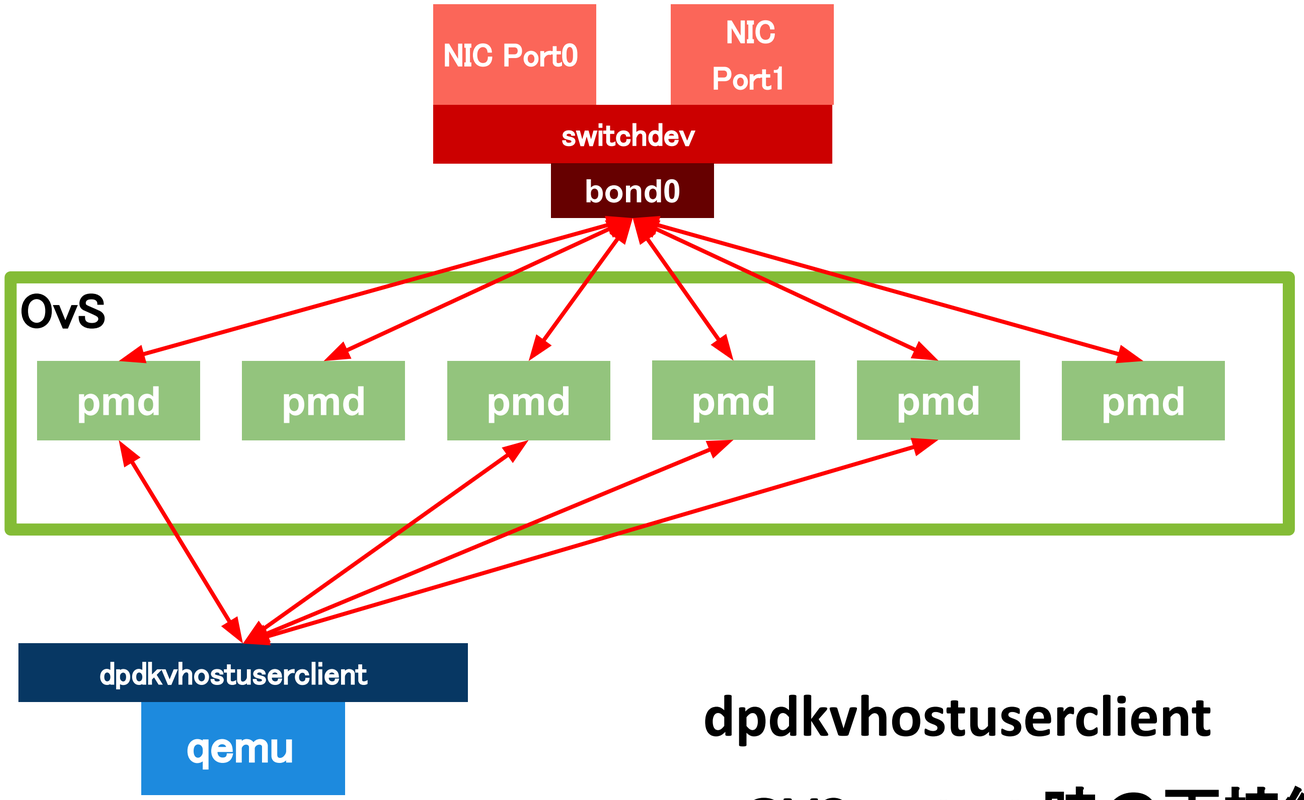


実現した OpenvSwitch 構成 (1)

- PMD 3cores 6threads
  - dpdkvhostuserclient
    - qemu をホストとして接続



実現した OpenvSwitch 構成 (2)

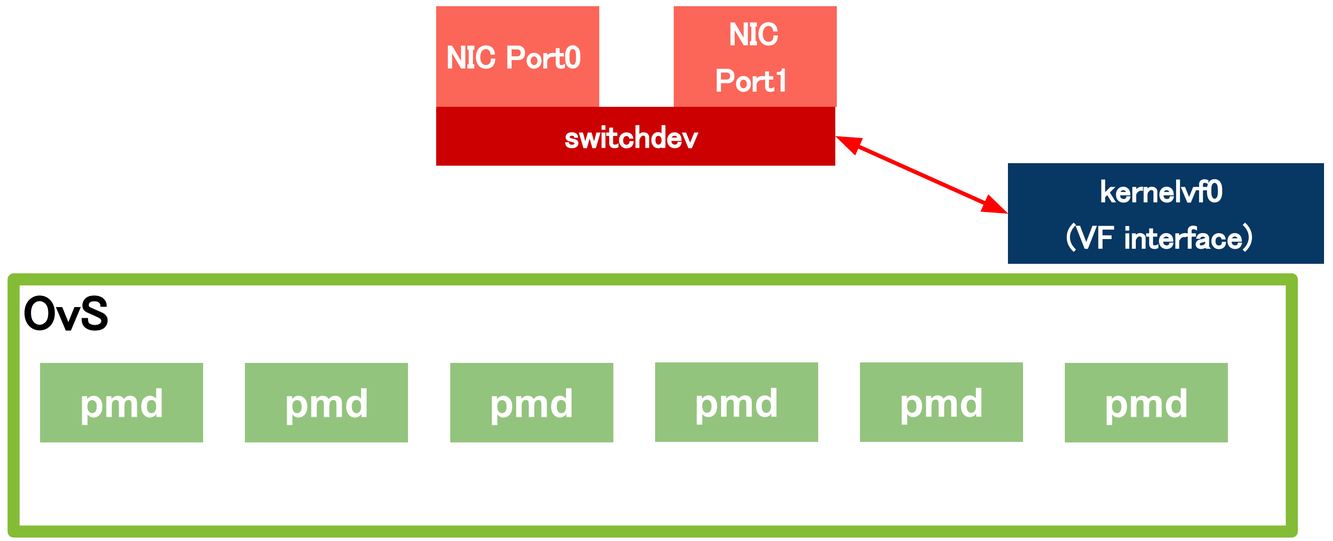


PMD: 3cores 6threads

dpdkvhostuserclient

OVS restart 時の再接続性を確保

ホスト側データパス構成



PMD: 3cores 6threads

- ConnectX-6 Dx の SR-IOV VF LAG の機能を利用
  - switchdev のレイヤでポート冗長される

OVS-DPDK チューニング

- そのままの DPDK 構成では 70Gbps 程度で頭打ったためチューニングを行った
  - PMD rxq pinning（bond0 のみ／100GbE x2 の受信処理を安定化）
  - pmd-rxq-assign: group（PMD 処理負荷を最適化しやすい）
  - TCP congestion control: BBR
  - NIC ring buffer 拡張
  - …

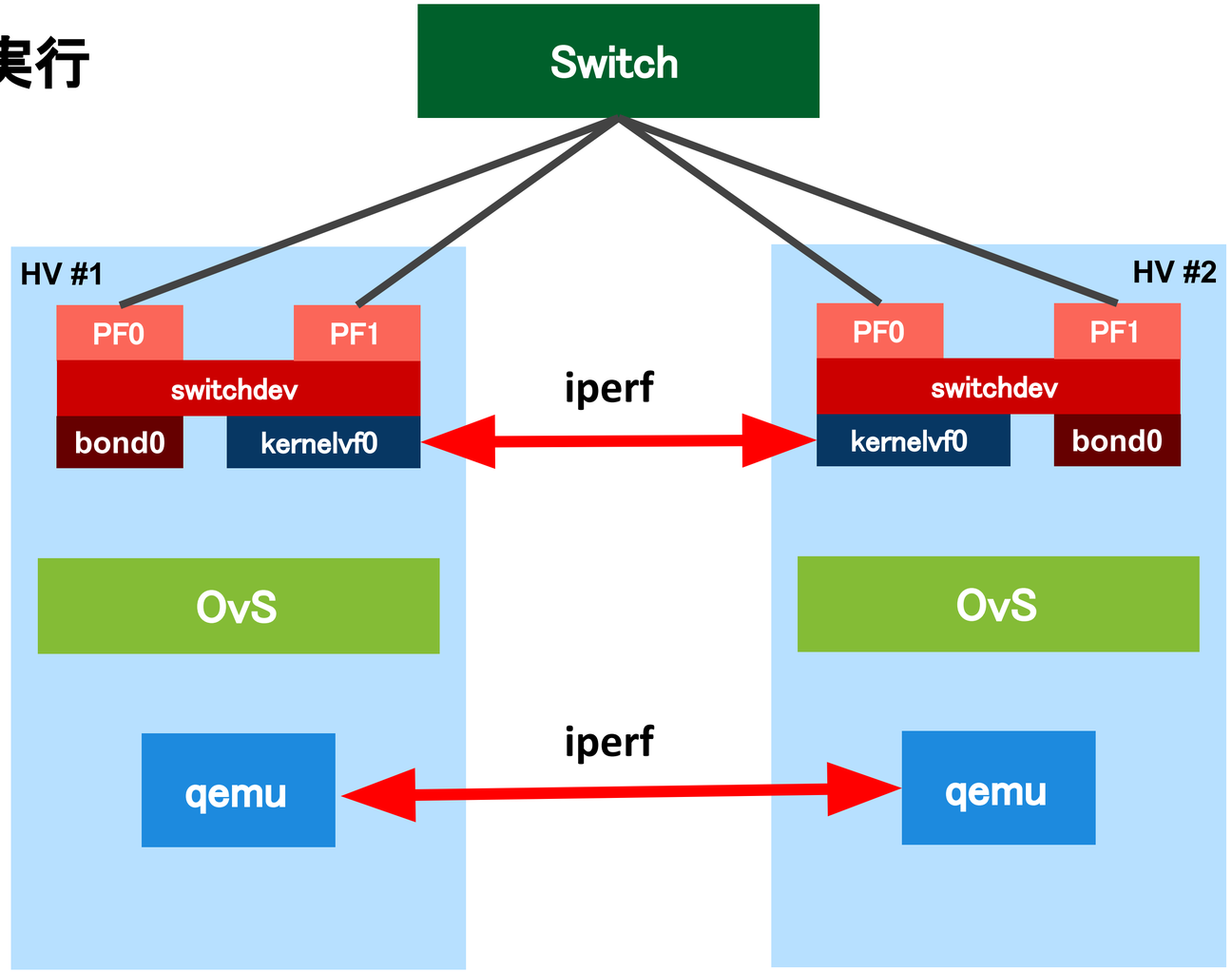
OVS-DPDK チューニング

- そのままの DPDK 構成では 70Gbps 程度で頭打ったためチューニングを行った
  - PMD rxq pinning（bond0 のみ／100GbE x2 の受信処理を安定化）
  - pmd-rxq-assign: group（PMD 処理負荷を最適化しやすい）
  - TCP congestion control: BBR
  - NIC ring buffer 拡張
  - …

素の DPDK 構成: 70Gbps
↓ tuning
200Gbps を使い切る構成を達成

性能測定

- ホスト側通信と VM 間通信で iperf を同時実行
- それぞれのスループットを合算し
  HV が処理可能な総帯域として評価
- 物理 NIC / bond / switchdev / OvS
  VM 通信を含む実運用に近い経路で測定

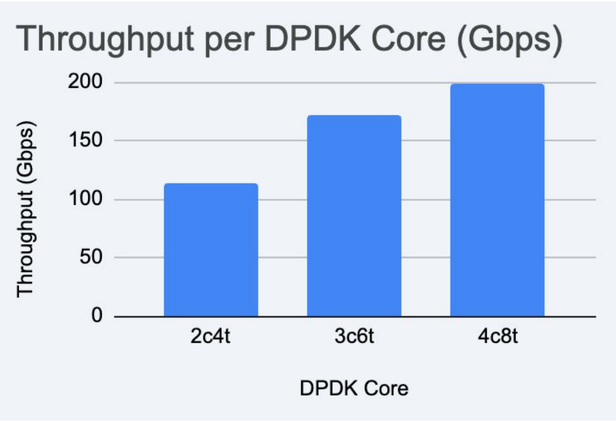


性能測定結果

- 目標: 2cores4threads で 200Gbps -> 実現できず
  - PMD に割く CPU を最小化、VM に最大限 CPU を残す

- 3cores6threads で 170Gbps
  - 200Gbps は達成可能だが、VM 収容効率とのバランスを重視

- 4cores8threads で 200Gbps を達成
  - OVS-DPDK 構成で物理帯域を使い切れることを確認



（再掲）既存 IaaS の性能と AWS

旧プライベートクラウド
(EPYC Rome)

Flavor	Core (thread)	Network Gbps
a1.large	2	1.2
a1.xlarge		
a1.2xlarge		
a1.3xlarge		
a1.4xlarge		

AWS 最新世代
(EPYC Turin)

Flavor	Core (thread)	Network Gbps
m8a.medium	1	0.52
		0.937
		1.875
		3.75
		7.5
m8a.8xlarge	32	15
m8a.12xlarge	48	22.5
m8a.16xlarge	64	30
m8a.24xlarge	96	40

80Gbps + 50Gbps + 40Gbps = 170Gbps ！

ストレージ :　　MAX 50Gbps
マイグレーション :　MAX 40Gbps

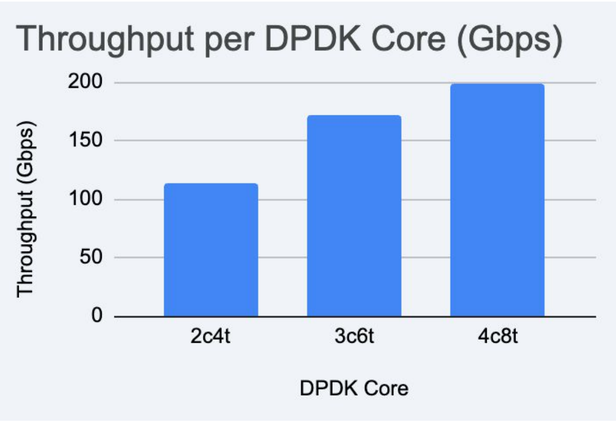
=> ハイパーバイザあたり　170Gbps の帯域が必要

性能測定結果

- 目標: 2cores4threads で 200Gbps -> 実現できず
  - PMD に割く CPU を最小化、VM に最大限 CPU を残す

- 3cores6threads で 170Gbps
  - 200Gbps は達成可能だが、VM 収容効率とのバランスを重視

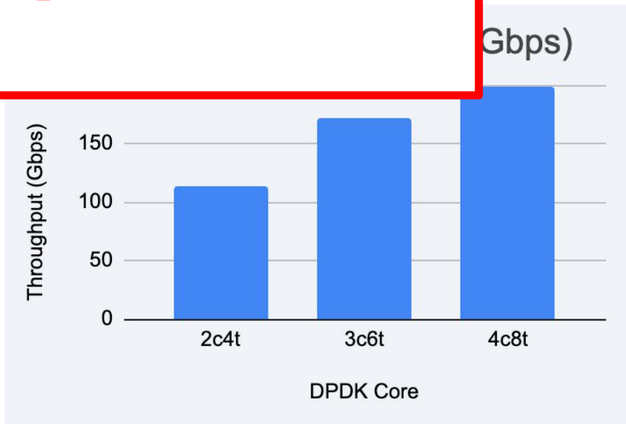
- 4cores8threads で 200Gbps を達成
  - OVS-DPDK 構成で物理帯域を使い切れることを確認



性能測定結果

- 目標: 2cores4threads で 200Gbps -> 実現できず
  - PMD に割く CPU を最小化、VM に最大限 CPU を残す
- 3c
  - 2
- 4cores8threads で 200Gbps を達成
  - OVS-DPDK 構成で物理帯域を使い切れることを確認

EPYC Turin の CPU パフォーマンスの高さにより
少ないコアでの高帯域構成を実現

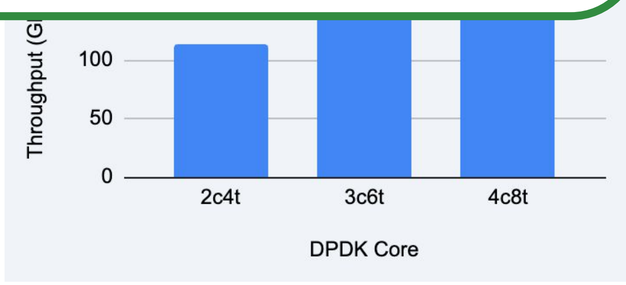


性能測定結果

- 目標: 2cores4threads で 200Gbps -> 実現できず
  - PMD に割く CPU を最小化、VM に最大限 CPU を残す
- 3c
  - 2
- 4cores8threads で 200Gbps を
  - OVS-DPDK 構成で物理帯域 切れることを確認

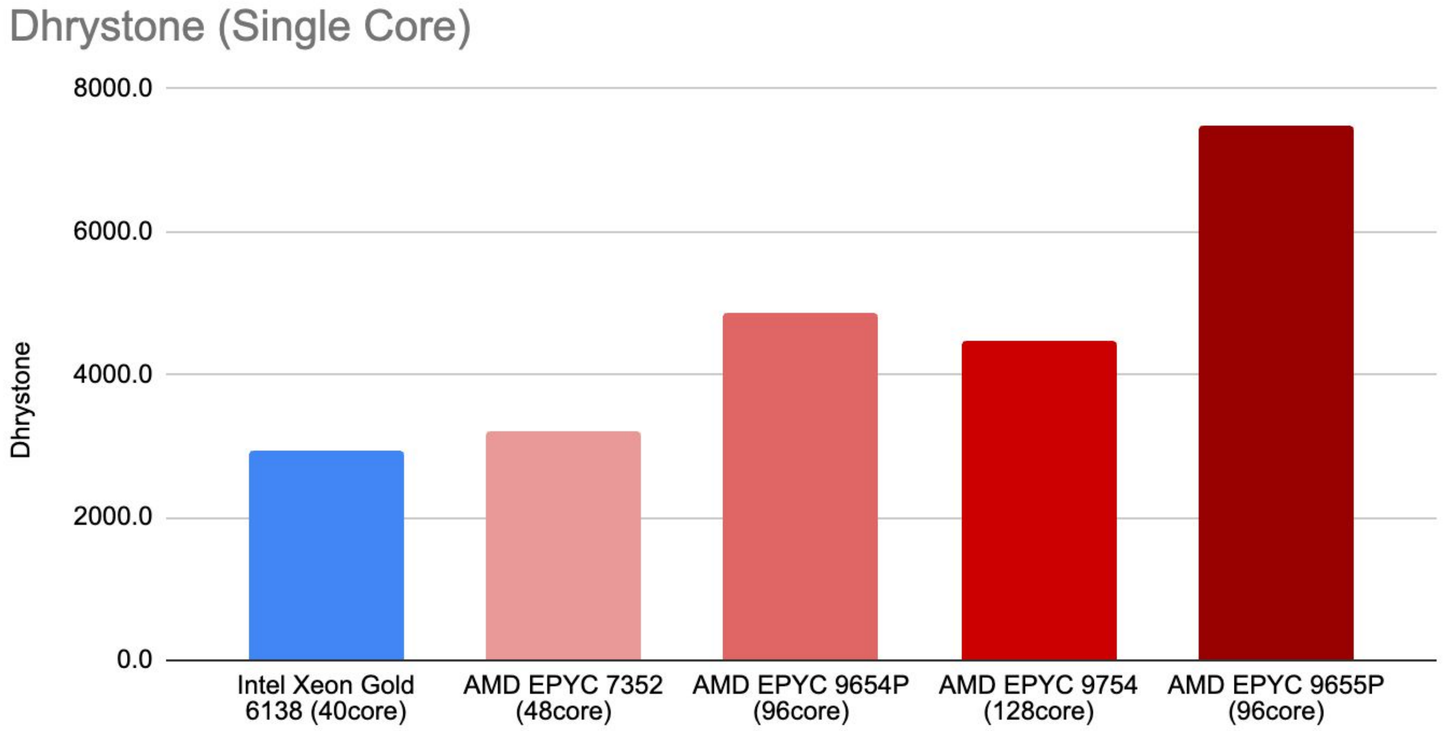
EPYC Turin の CPU パフォ
少ないコアでの高帯

他の CPU だとどうだった？



他の CPU での DPDK 性能 (1)

- DPDK PMD 処理性能は CPU の single-thread 性能・整数演算性能の影響を大きく受ける
  -> Dhrystone スコアを PMD 必要コア数の概算指標として利用



他の CPU での DPDK 性能 (2)

- 既存のプライベートクラウド　CPU
  - Intel Xeon Gold 6138　　:　8cores 16threads
  - AMD EPYC 7352　　:　7cores 14threads
- 採用検討 CPU
  - AMD EPYC 9654P　　:　5cores 10threads
  - AMD EPYC 9754　　:　6cores 12threads
- 採用 CPU
  - AMD EPYC 9655P　　:　3cores 6threads

他の CPU での DPDK 性能 (2)

- 既存のプライベートクラウド　CPU
  - Intel Xeon Gold 6138　　:　8cores 16threads
  - AMD EPYC 7352　　:　7cores 14threads
- 採用検討 CPU
  - AMD EPYC 9654P　　:　5cores 10threads
  - AMD EPYC 9754　　:　6cores 12threads
- 採用 CPU
  - AMD EPYC 9655P　　:　3cores 6threads

EPYC Turin の CPU パフォーマンスの高さが際立つ

6

まとめ

CyberAgent

まとめ

- パブリッククラウドと戦うためには、プライベートクラウドにおいても
  高帯域なハイパーバイザネットワークが必要
- OVS-DPDK にて 200Gbps ready なハイパーバイザ基盤を実現
  - 実運用設計は 170Gbps
- EPYC Turin により、少ない PMD コアでも 170Gbps 級の実運用性能を実現

7

今後の展望

CyberAgent

次期 HV ネットワークに向けて

- NVIDIA NIC の推奨・サポート範囲が DPU / vDPA / SR-IOV よりに変化している
  - 最新の NVIDIA の Software Framework では DPDK が削除される等

- DPU: BlueField 3 で NIC 上で OVS / virtio を動かす
  - Pros: HV から仮想ネットワーク処理を分離できる
  - Cons: ベンダーロックインやコスト
- vDPA: ConnectX NIC で vDPA 構成
  - Pros: 比較的安価に低遅延・大容量な仮想ネットワークを実現できる
  - Cons: OpenStack へのインテグレーションコスト

EOP

参考資料

- **Partial Offload Optimization and Performance on Intel Ethernet 700 Series NICs Using rte_flow**

  https://www.openvswitch.org/support/ovscon2019/day1/1305-OVS_Conference_OVS-DPDK_Partial_Offload.pdf