Rethinking AI Infrastructure: LINEヤフーが描く、内製技術で切り開くネットワークとエンジニアリングの新時代

LINEヤフー株式会社

大浦 晋

道下 幹也

LINEヤフー

登壇者紹介

Speakers

- 大浦 晋 (Shin Oura)
- ネットワーク本部 ネットワーク1部 サービスネットワーク1チーム
- GPU NWの設計、構築、運用を担当

- 道下 幹也 (Mikiya Michishita)
- クラウドインフラストラクチャ本部 IaaS開発部 Computeチーム
- GPUコンピュート基盤の設計や技術検証・開発を担当

Agenda

01

LINEヤフーにおけるAIインフラとその課題

02

Rethinking AI Infrastructure

03

ACP強化における具体的な技術選定

04

実際に設計・構築してどうだったか？

05

新たに見えてきた課題と今後の取り組み

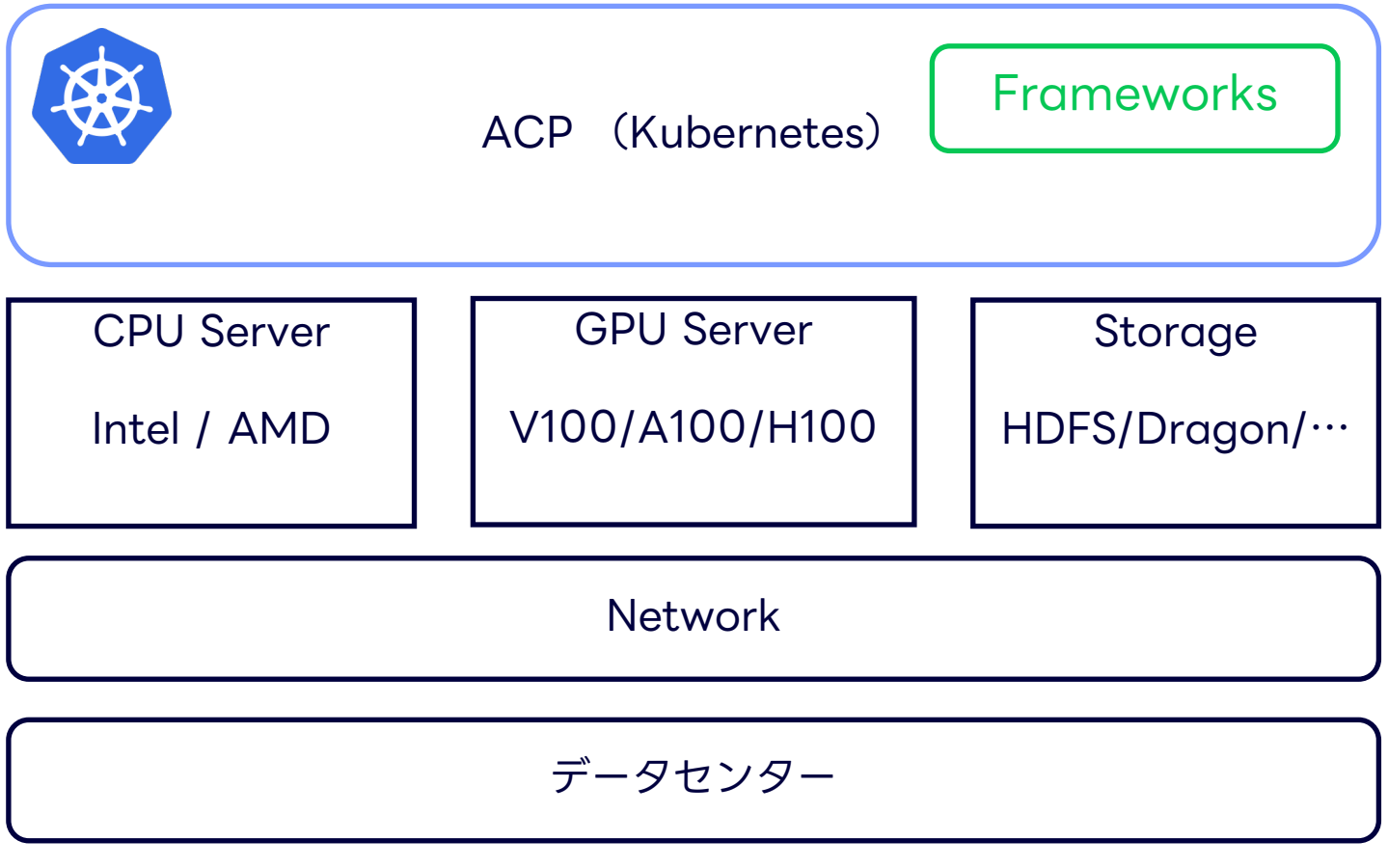
06

まとめ・議論

LINEヤフーにおける
AIインフラとその課題

LINEヤフーのAI Cloud Platform（ACP）

LINEヤフーのAI開発者向けプラットフォーム

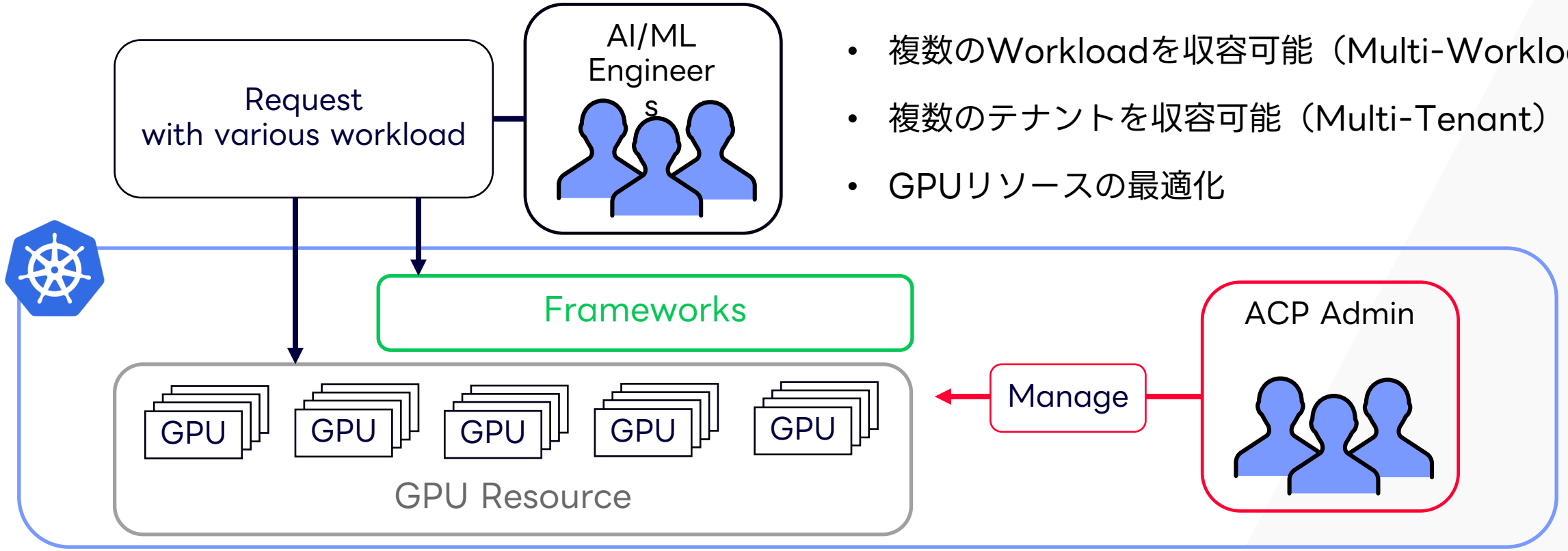


全ての領域を自社*で構築、運用

* データセンターはActapio, Inc.（LINEヤフーの100%子会社）が構築・運用

ACPを支えるインフラに求められる要件

PrivateなPlatformならではの技術要件と課題



- 複数のWorkloadを収容可能（Multi-Workload）
- 複数のテナントを収容可能（Multi-Tenant）
- GPUリソースの最適化

LINEヤフーにとってのAIインフラ

なぜオンプレのAIインフラを持っているのか

コストメリット

- パブリッククラウドに対して費用が安い
- 構成を自分たちで決定でき、コストコントロールがしやすい

柔軟性

- ワークロードに合わせた調整が容易
- 運用経験を通じたインフラ面の最適化が行いやすい

データの取り扱い

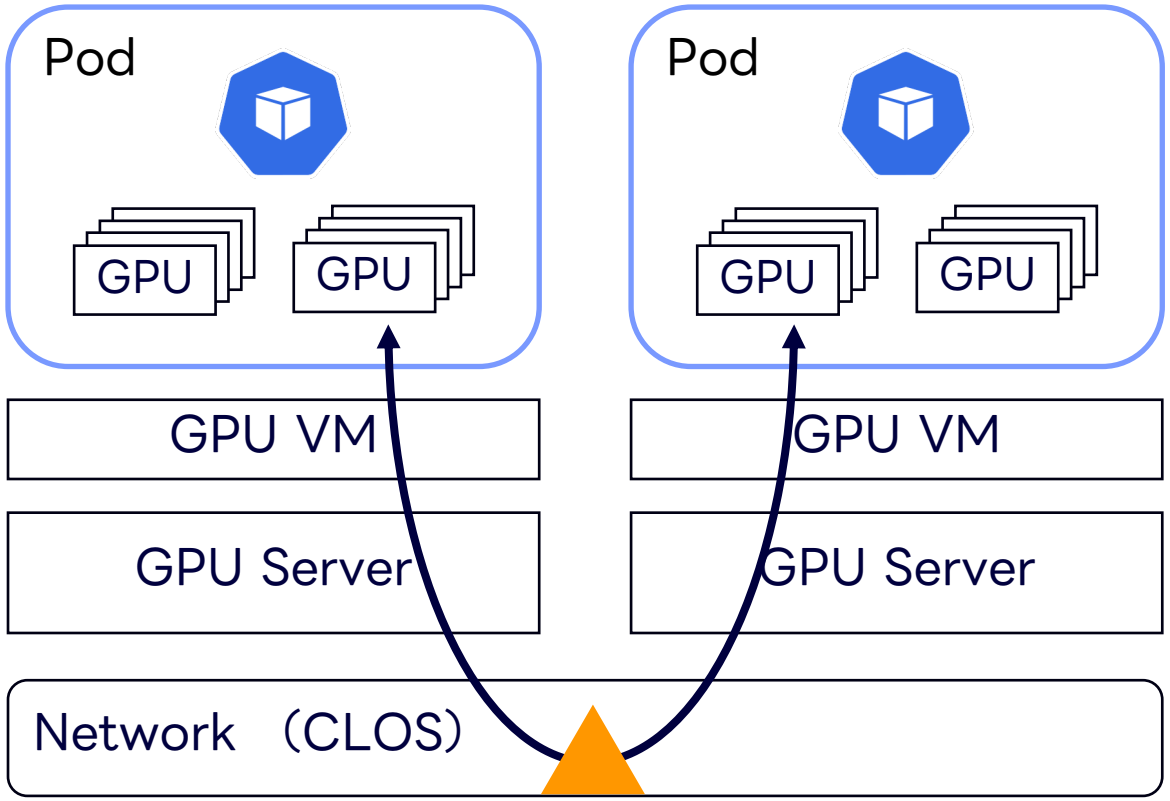
- 社内規定に即した学習データ等の取り扱いがしやすい

ナレッジの獲得

- DCからアプリケーションのレイヤまで、AI／MLのナレッジが獲得可能

ACPを支えるインフラの課題

分散学習実行時の性能の向上



GPUサーバを跨ぐ通信（分散学習）
で十分な通信性能が出ない

近年は分散学習が一般的な世界
分散学習需要に応える必要がある

将来的なビジョンと方向性

既存の課題に対して、どのようなビジョンを持ち、どのような方向性を定めて進めたか

解決したい課題と要求事項

ACPを分散学習にも適した環境として強化する

- 複数ワークロード、ユーザ、テナントの収容
- 高速な分散学習向けGPU NWを提供する
- 既存資産を活用し安定した運用を行う

ビジョン

進化していく世界に遅れない
「インフラ」と「人材」を作る

方向性

Rethinking AI Infrastructure

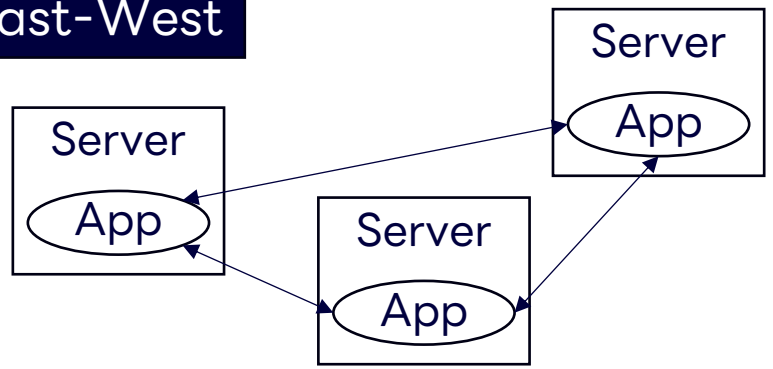
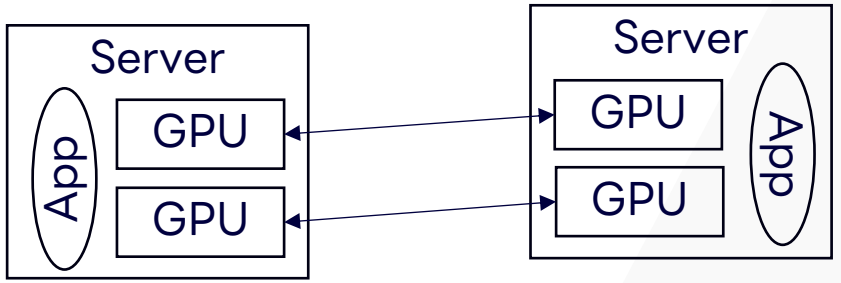
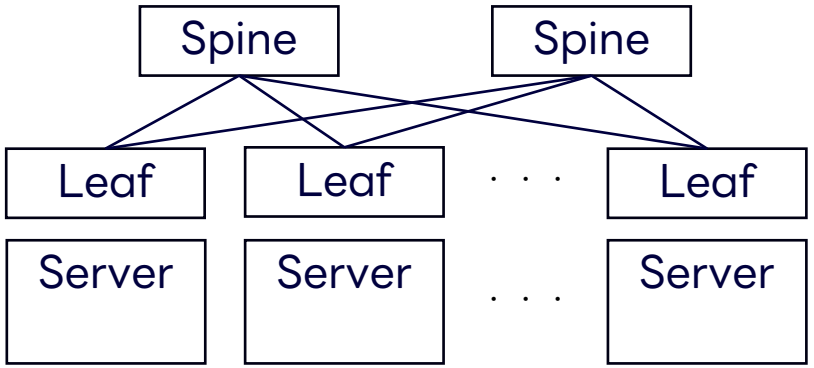
（詳細は後述）

- Networkのみでなく全体最適化を基軸として設計
- 適切な技術の選択、時には内製開発をし、自分たちのインフラに最適な形を常に模索する

Rethinking AI Infrastructure

AI関連技術の飛躍とインフラの変化

生成AIに代表される技術の発展が世界を一変させ続けている

	Current Datacenter Traffic	*GPU Traffic*
Traffic Pattern	East-West 	East-West (GPU Optimized) 
Architecture	CLOS 	Many Architectures are introduced • Rail-Optimized Topology • Rail-Only Topology • Dragonfly Topology …etc

世界の転換の捉え方

3つの視点から見る世界の転換

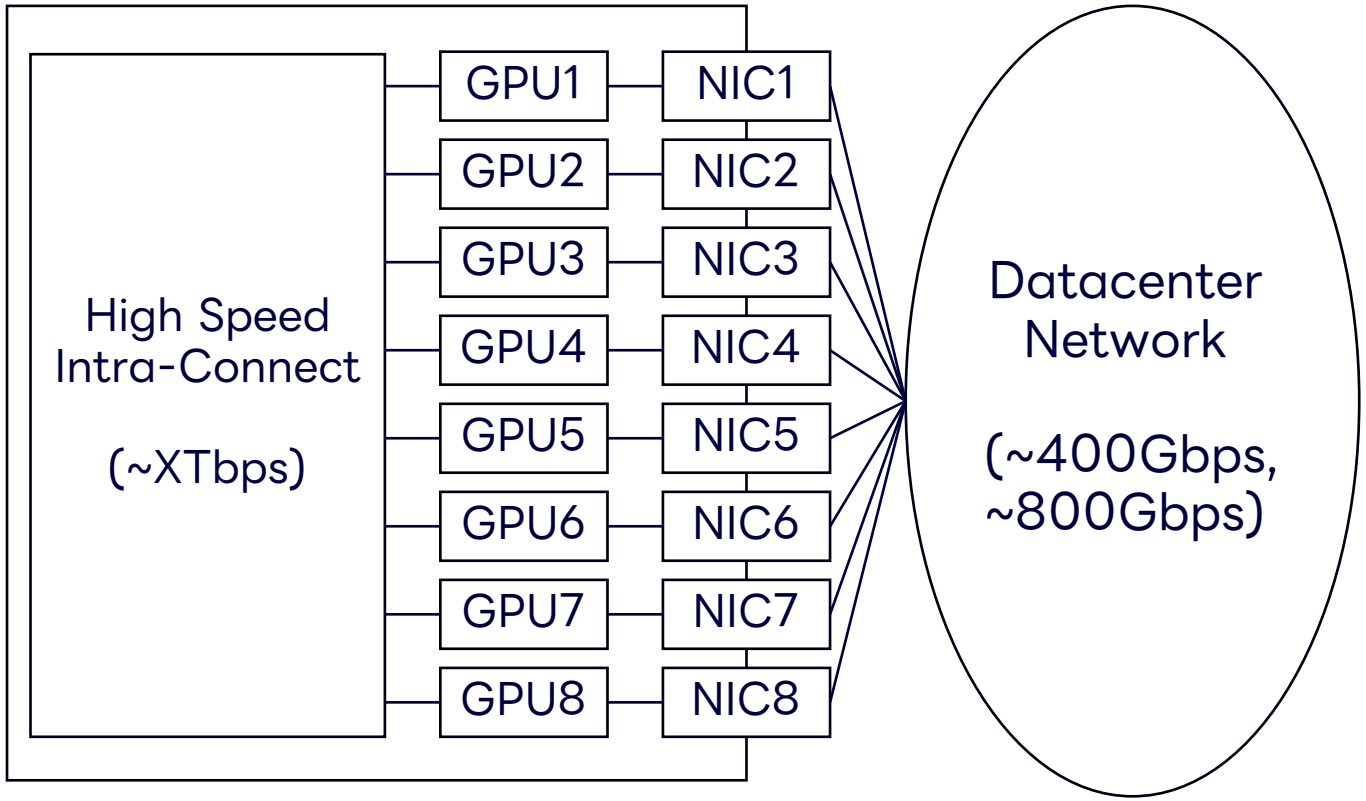
技術の転換

運用の転換

人材の転換

技術の転換

ネットワークとプラットフォームの運用のあり方を考え直す必要性



- NICが200/400/800GbE, 1.6TbEと驚愕的な速度で進化中
- サーバレイアウトの変化に伴い、Networkデザインも変化
- Inter-Connect >> Intra-Network
  まだまだNWはボトルネック

プラットフォーマはHW以外のボトルネックにも目を向け、それを取り除くための最大限の努力が必要

運用の転換・人材の転換

ネットワークとプラットフォームの運用のあり方を考え直す必要性

運用

Server Topologyの理解

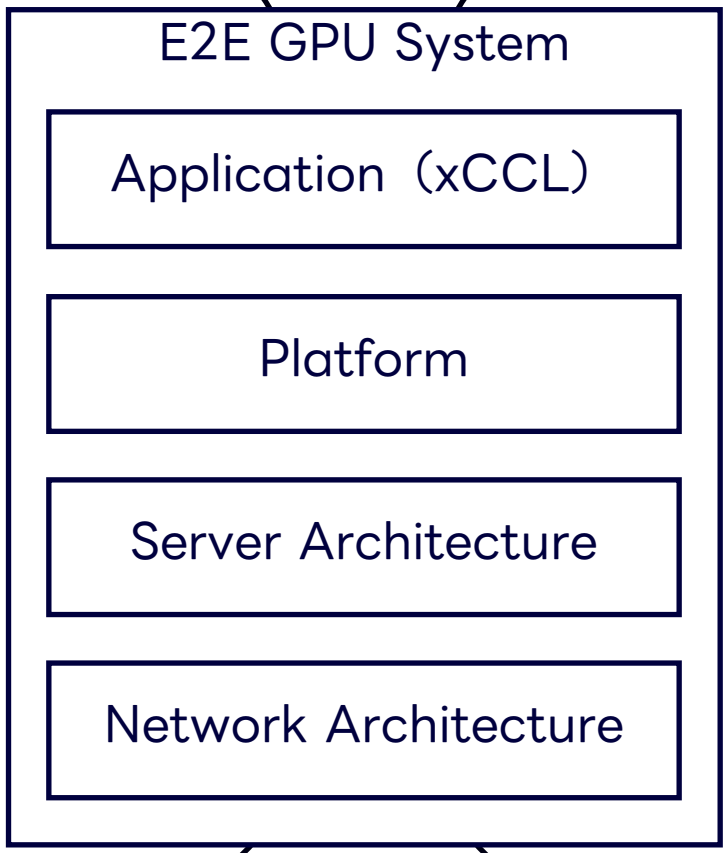
xCCLの理解

↓

高性能なNWの設計、運用

GPU通信の詳細な議論

が可能になる



人材

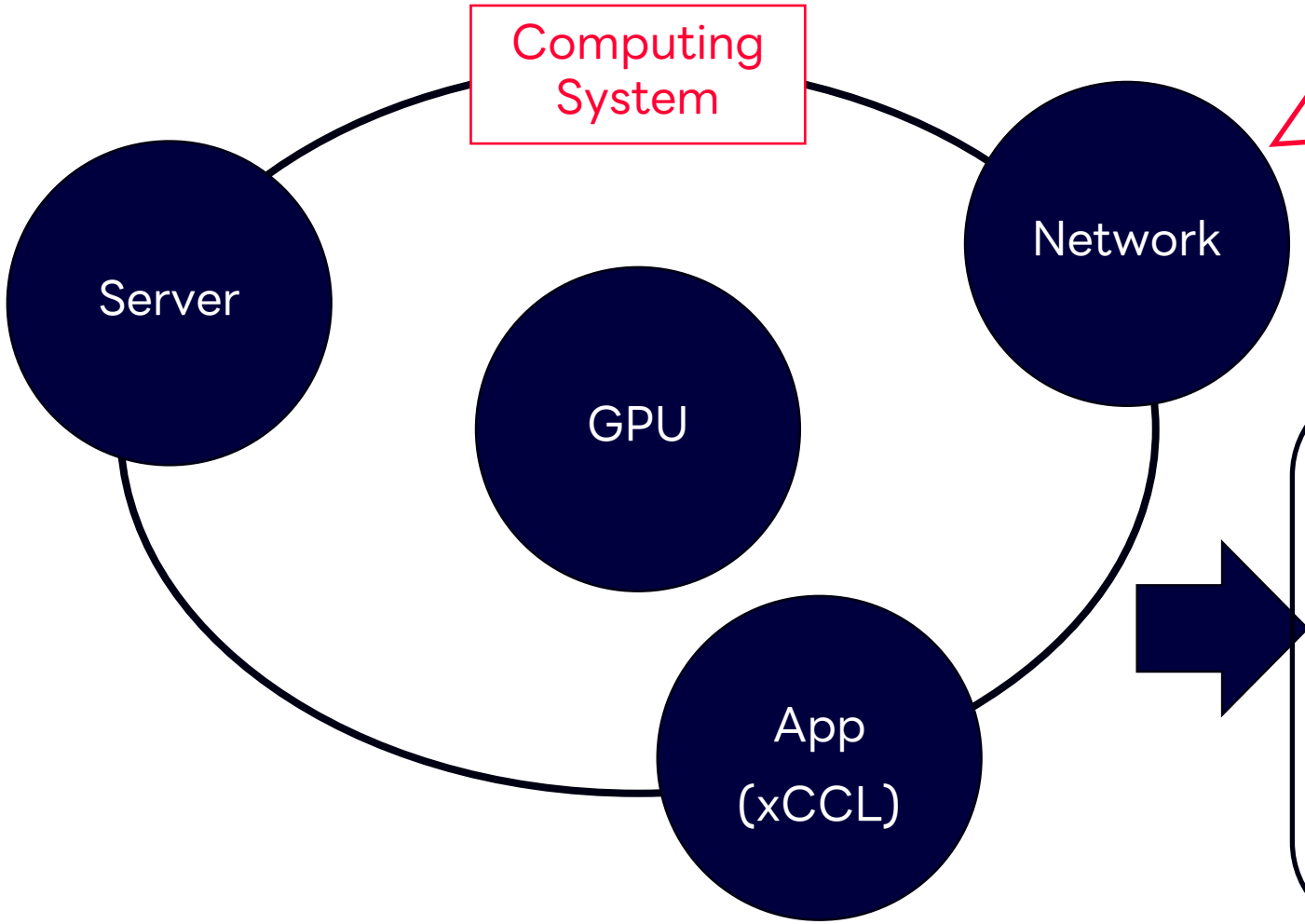
Networkの領域のみでなく、

ServerやxCCLなどの領域の

スキルセットを獲得していける人材が重要

GPU-Centric Infrastructure

GPUのための基盤を作るために考え方を転換する



Networkは
コンピューティングシステムの一部

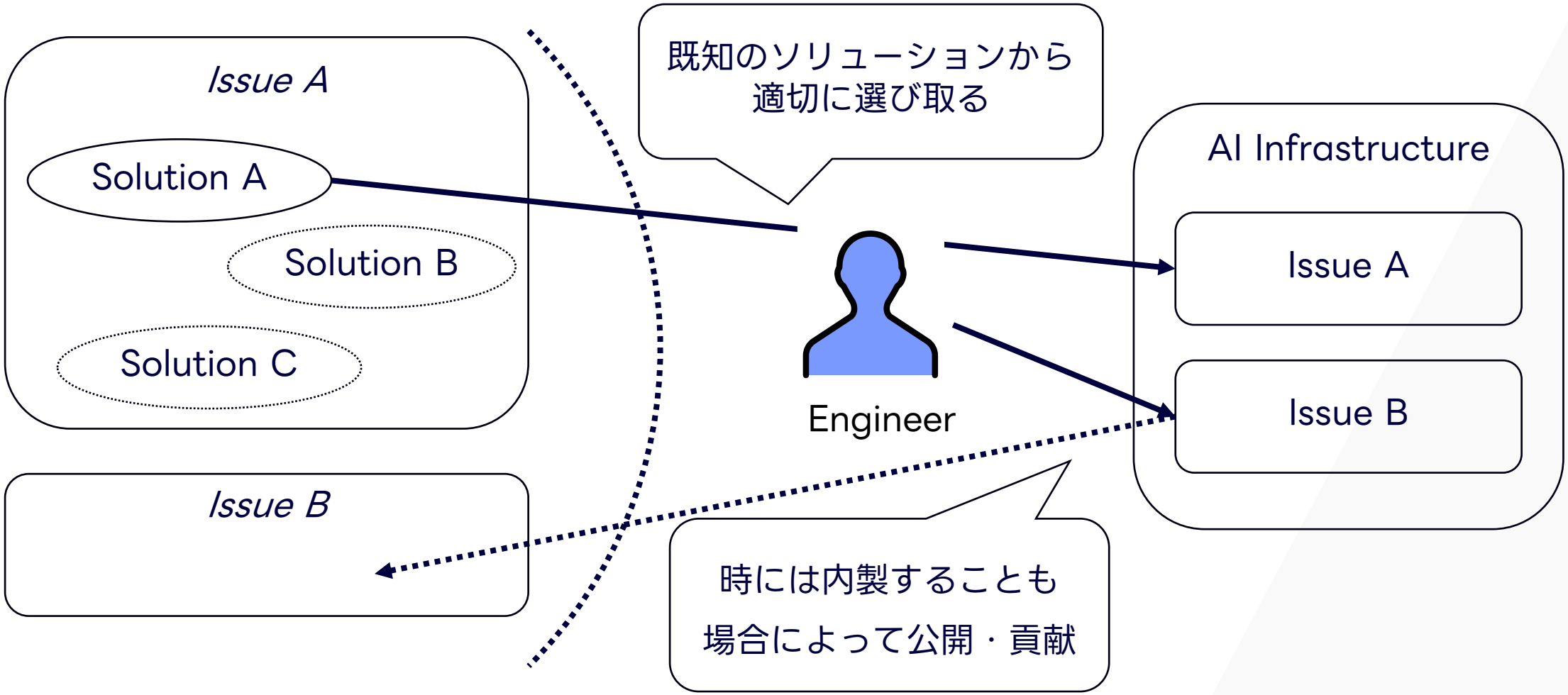
GPUを中心として取り巻くコンピューティングシステム全体の最適化が必須

+

「自分たちのインフラには何が必要か」をエンジニアとして判断する力が必要

時代の転換期とエンジニアリング

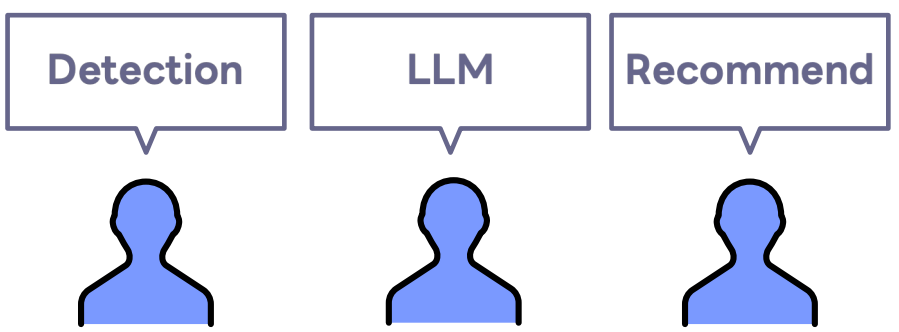
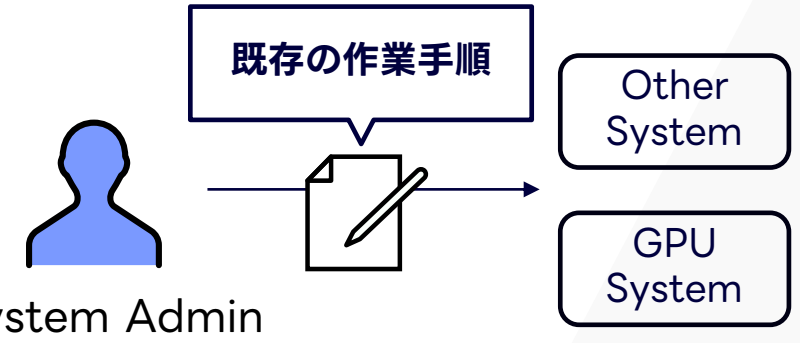
自社のインフラに必要な技術の取捨選択を行う能力の重要性



ACP強化における
具体的な技術選定

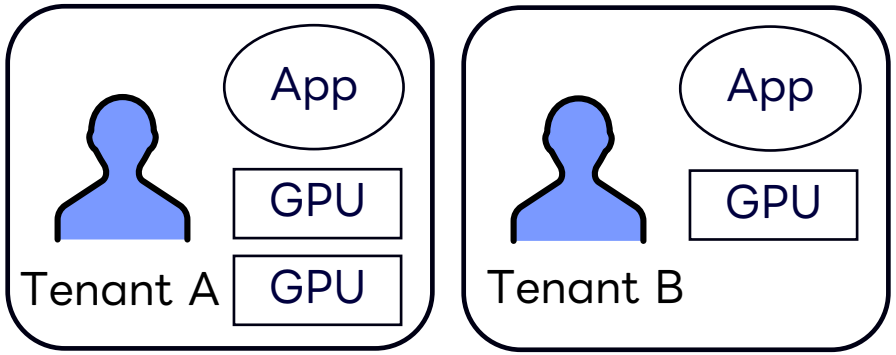
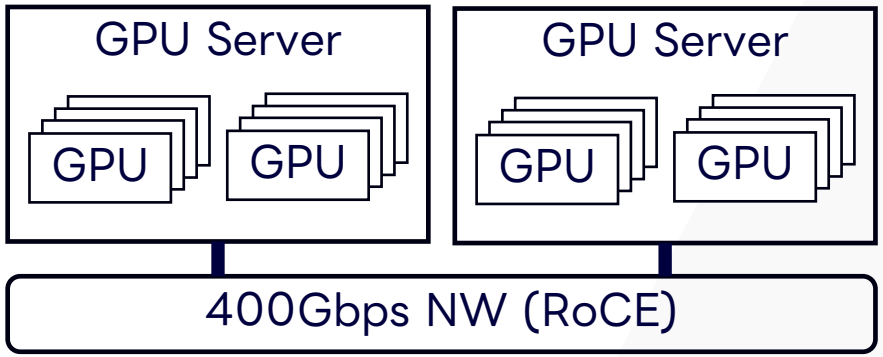
課題の解釈

具体的な要求と選択したソリューション

	Multi-Workload（割愛）	*運用互換性・既存資産の流用*
課題・要求事項		
技術的要求 考慮事項	• リソース管理・スケジューリング • フレームワークなどのサポート	• 構成差分の最小化 • 管理NWの冗長
我々の選択	• スケジューリング方式の検討 • AI/ML Framework サポート, …etc	• BGPベースのNW構成 • BGPベースの管理NWの冗長

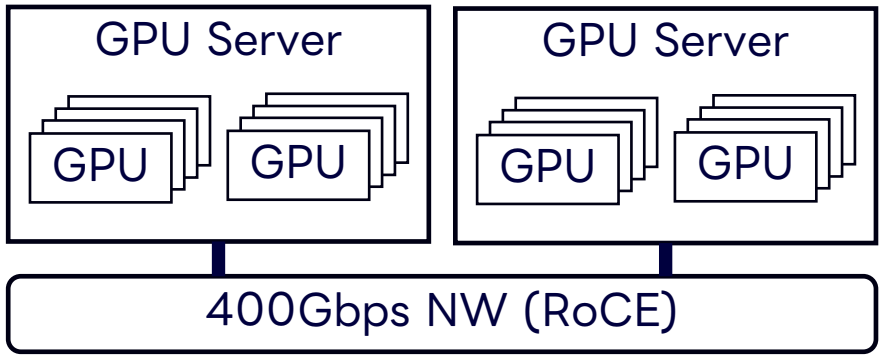
課題の解釈

具体的な要求と選択したソリューション

	Multi-Tenancy	*分散学習向けNW*
課題・要求事項		
技術的要求 考慮事項	• プラットフォーム上のテナント分離 • パフォーマンス影響のないNW分離	• GPU NWの構成 • 仮想化の廃止
我々の選択	• k8s namespaceによるテナント分離 • OpenvSwitchによるNW分離	• GPU NWの構築 • サーバ構成・サーバ内部NWの構成

GPU NWの構築

議論のスコープ

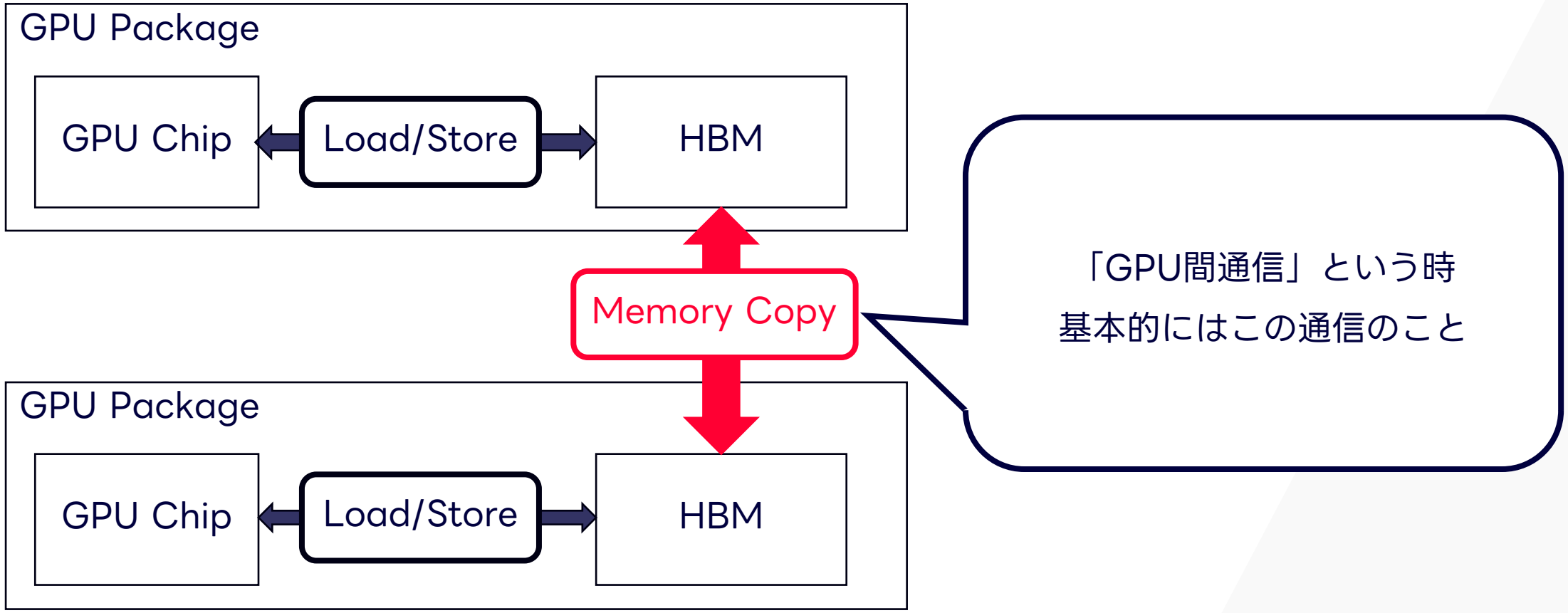
	分散学習向けNW
課題・要求事項	
技術的要求 考慮事項	• GPU NWの構成 • 仮想化の廃止
我々の選択	• GPU NWの構築 • サーバ構成・サーバ内部NWの構成

以降の議論の流れ

- 「GPU間通信」とは何か？
- GPUのためのネットワークとその選択
- GPU間通信の特性とNW Topology

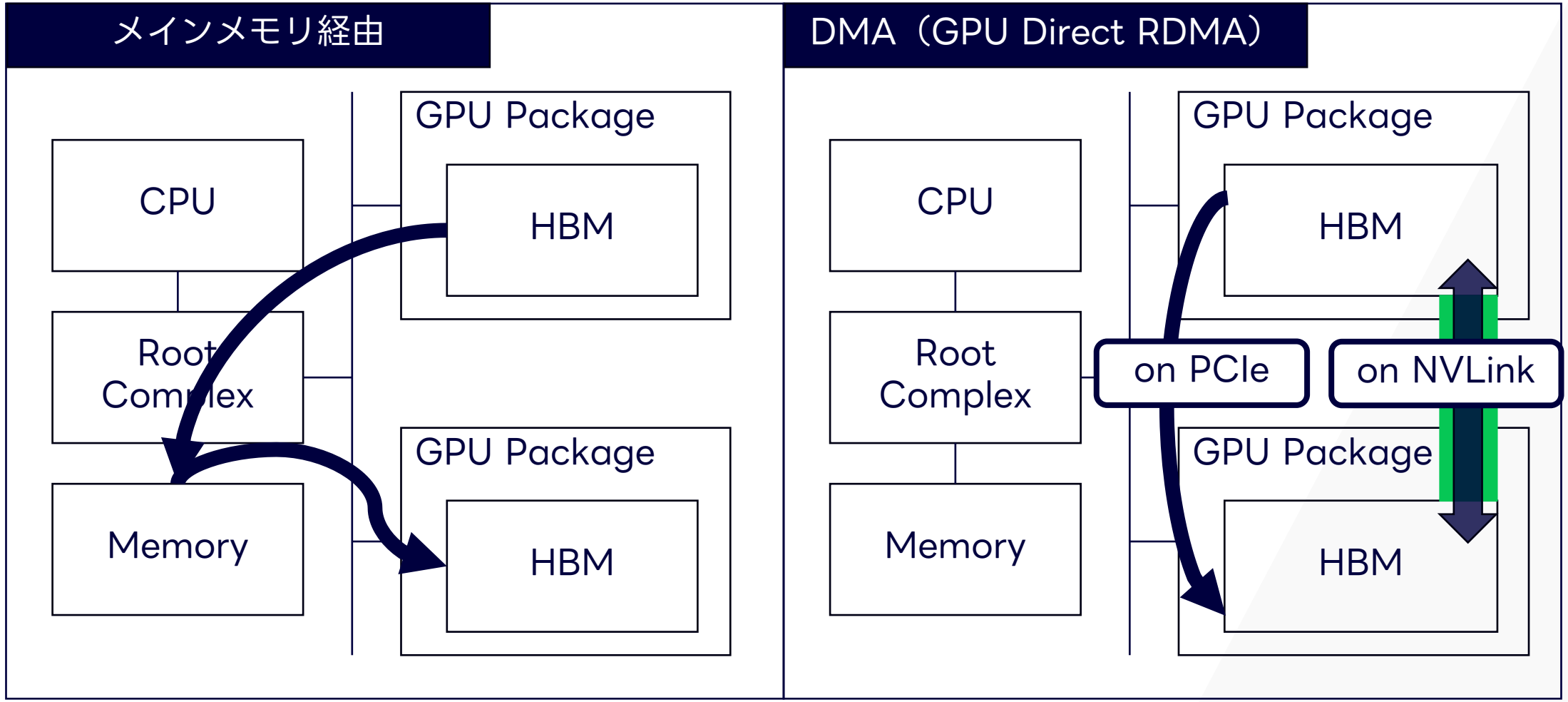
GPU NWに入る前に

GPU間通信とは何を指しているか？



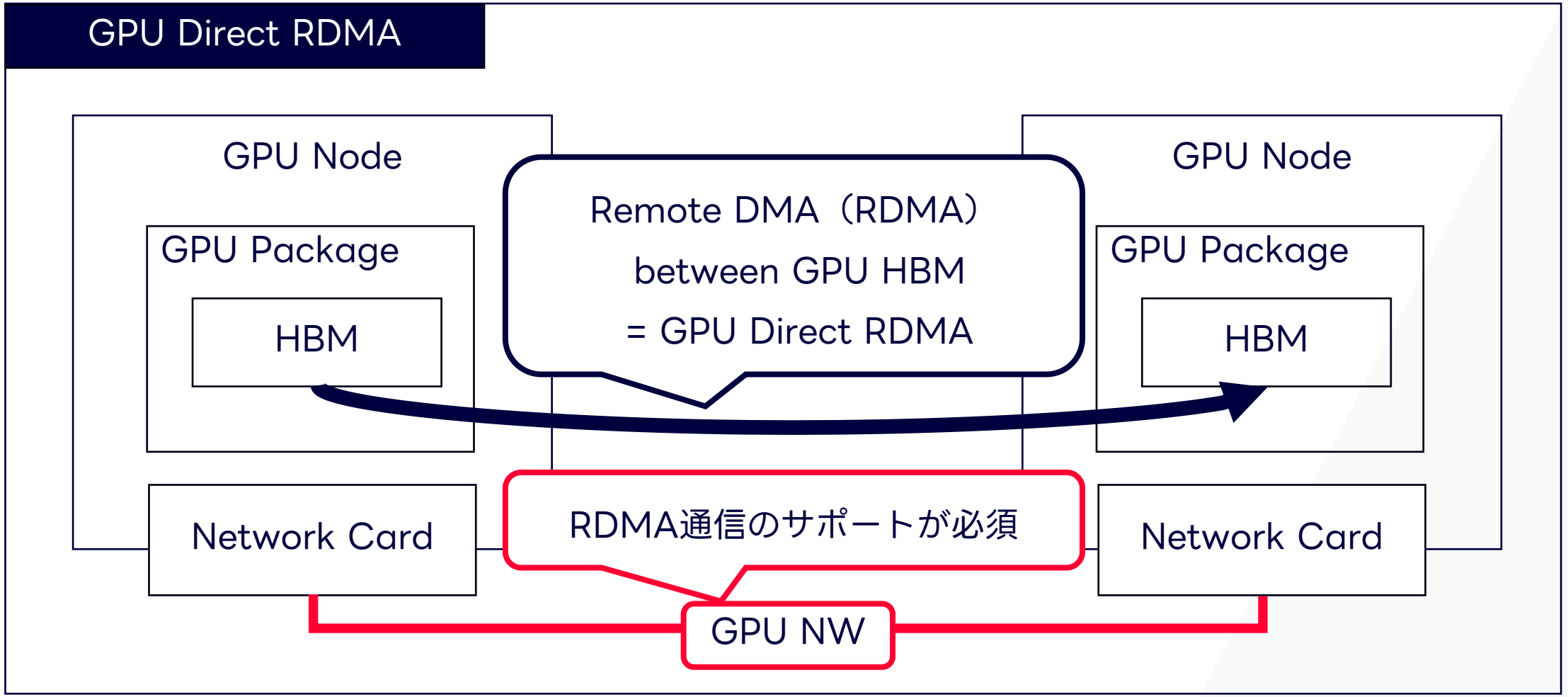
GPU間のデータコピー方式

メインメモリ経由のデータコピーとGPU Direct P2P（DMA）



異なるノードのGPU間通信

GPU Direct RDMAと、NWに対するRDMAの要請



課題の解釈

RDMAの要件

低遅延

大容量通信

ロスレス

GPUのためのネットワーク

InfiniBand vs Ethernet

InfiniBand

- HPCで長年の運用実績
- 高速・低遅延・高信頼性
- 原理的にロスレス
- 業界団体で仕様策定
- 専用ハードウェアに実装

RoCEv2（Ethernet）

- インターネットベースの運用実績
- 回復力のある上位層のトランスポート
- パケットロス前提の設計
- コミュニティで仕様策定(オープン)
- コモディティな実装

LINEヤフーではInfiniBandとEthernetの両方でGPU基盤を運用

- 特定の用途の小規模環境でInfiniBandを採用
- **一定以上の規模のクラスタではEthernetを採用**しコストパフォーマンスを最適化

なぜ Ethernet を選択するのか

技術的課題解決と人材育成のサイクルをつくることが我々の価値

Multi-Tenancy

- クラウド基盤としてのテナント分離
- 異なるワークロードエンティティの収容
- セキュリティ要件

内製開発

- 自社のワークロードに応じた開発
- パフォーマンスチューニング
- オブザーバビリティ

人材育成

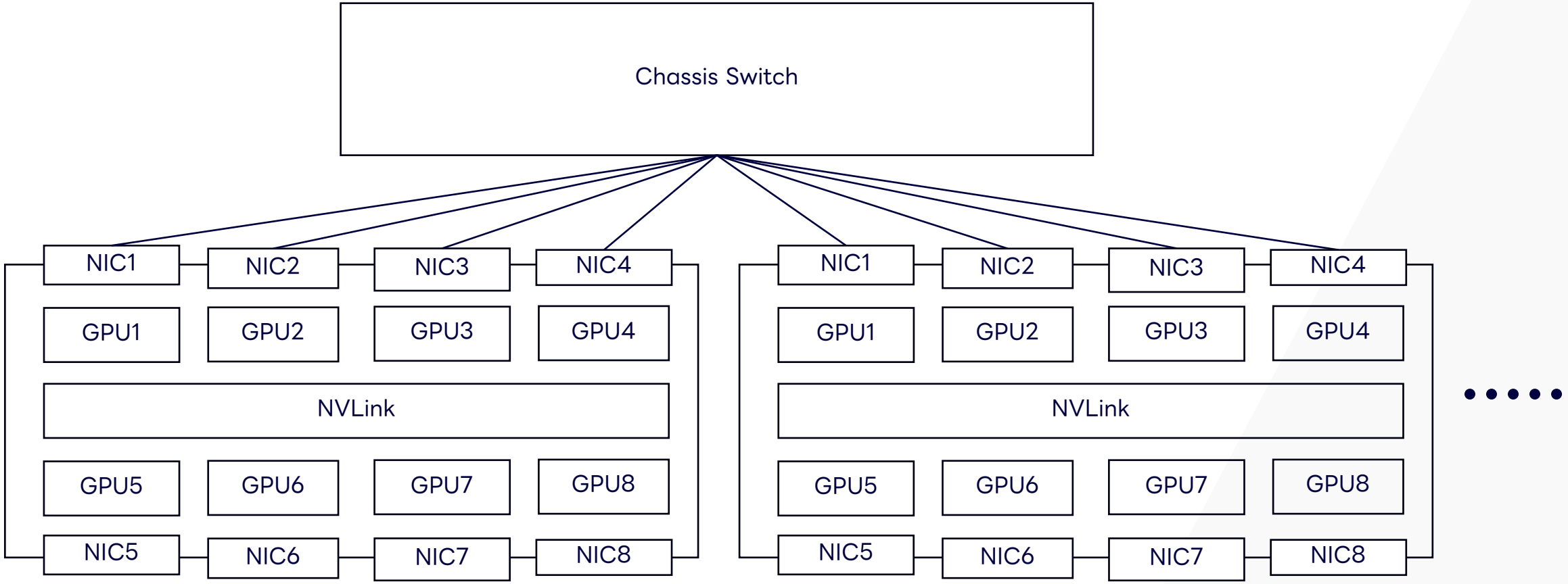
- AIインフラを内製できる人材育成
- 先端技術の知見の蓄積
- 運用スキルの継承

オープン化

- Ethernetの豊富な知見と運用実績
- 仕様と実装が広く公開されている
- 業界へのフィードバックと貢献

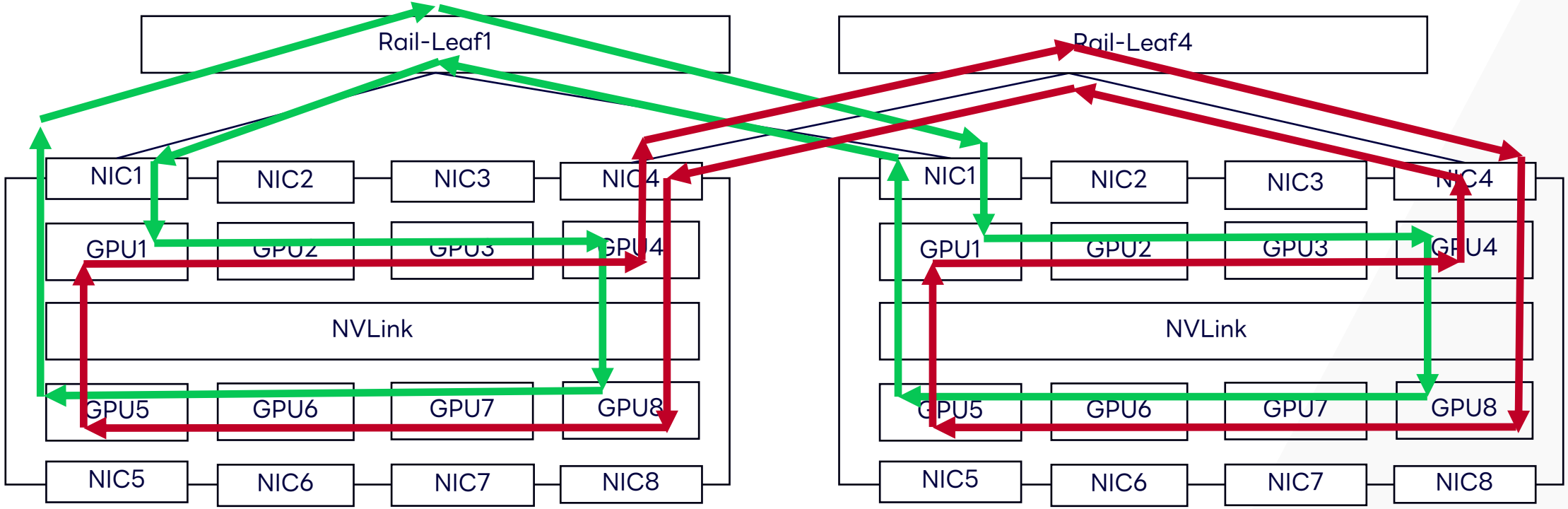
Single Chassis Network

集合通信の性能を最適化しつつ、スケールアウト可能にする構成



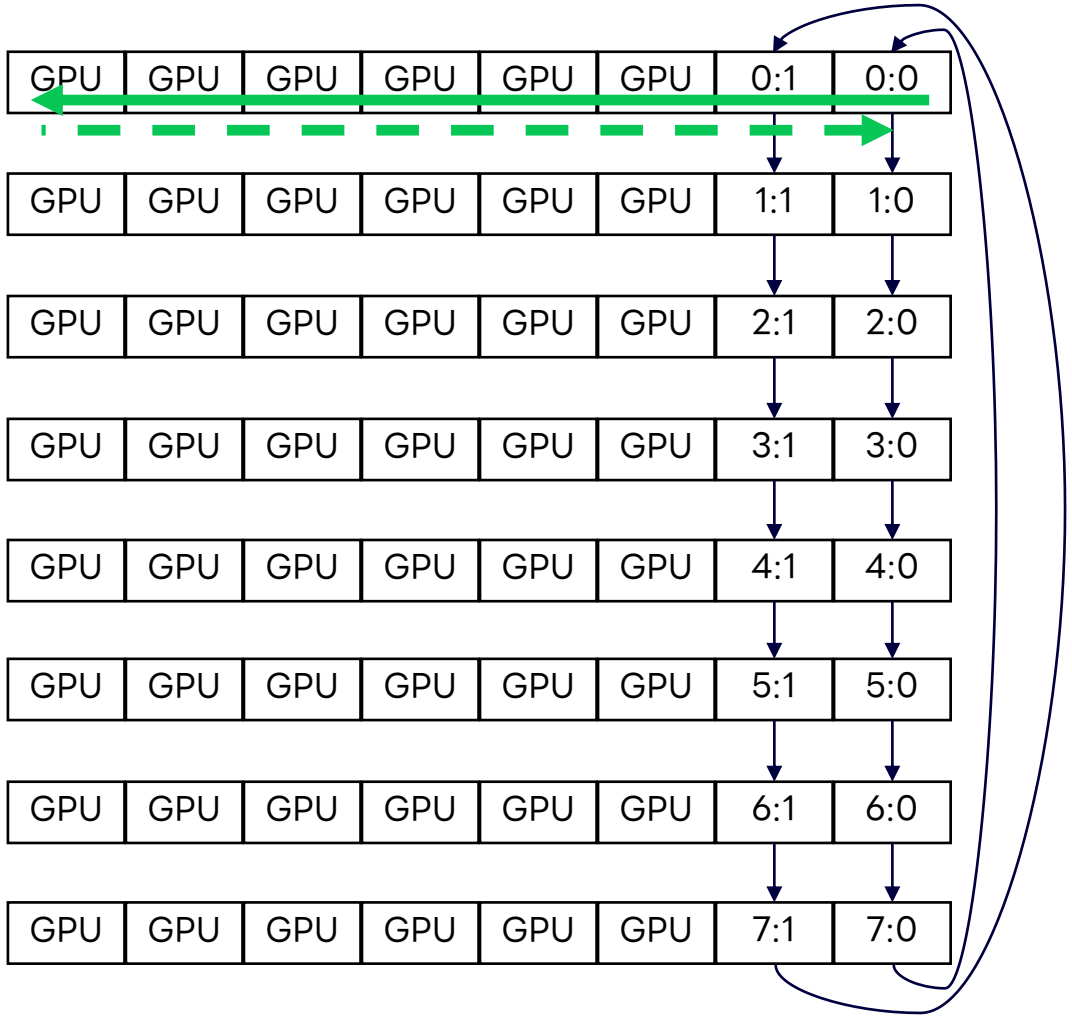
Railの導入と集合通信

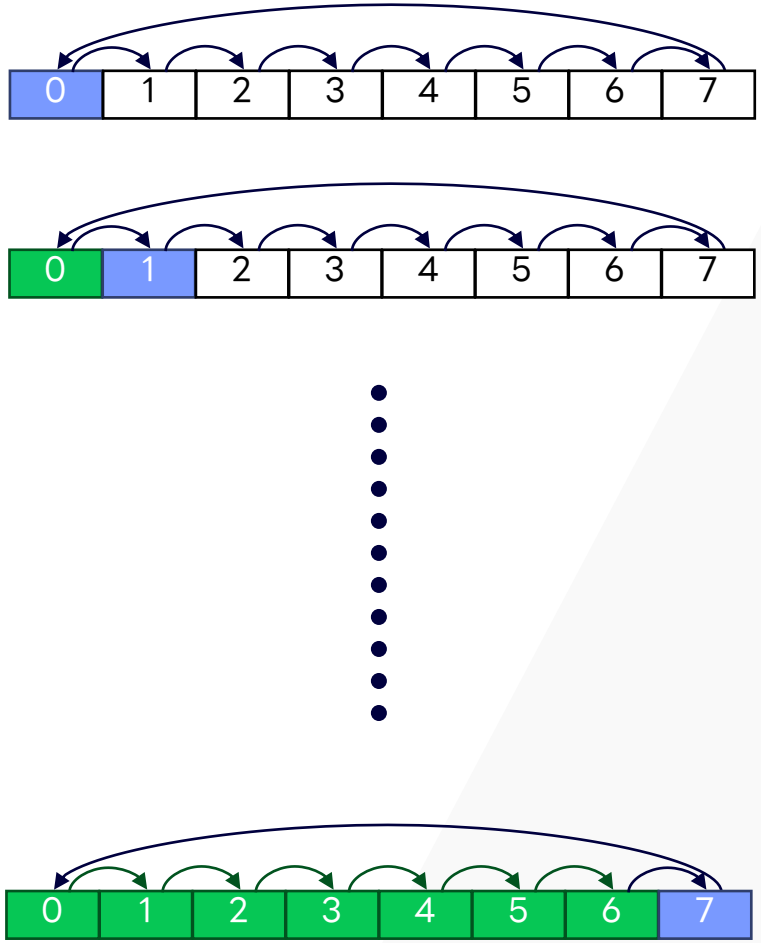
集団通信のアルゴリズムを最適化するトポロジー構成



Ring Algorithm

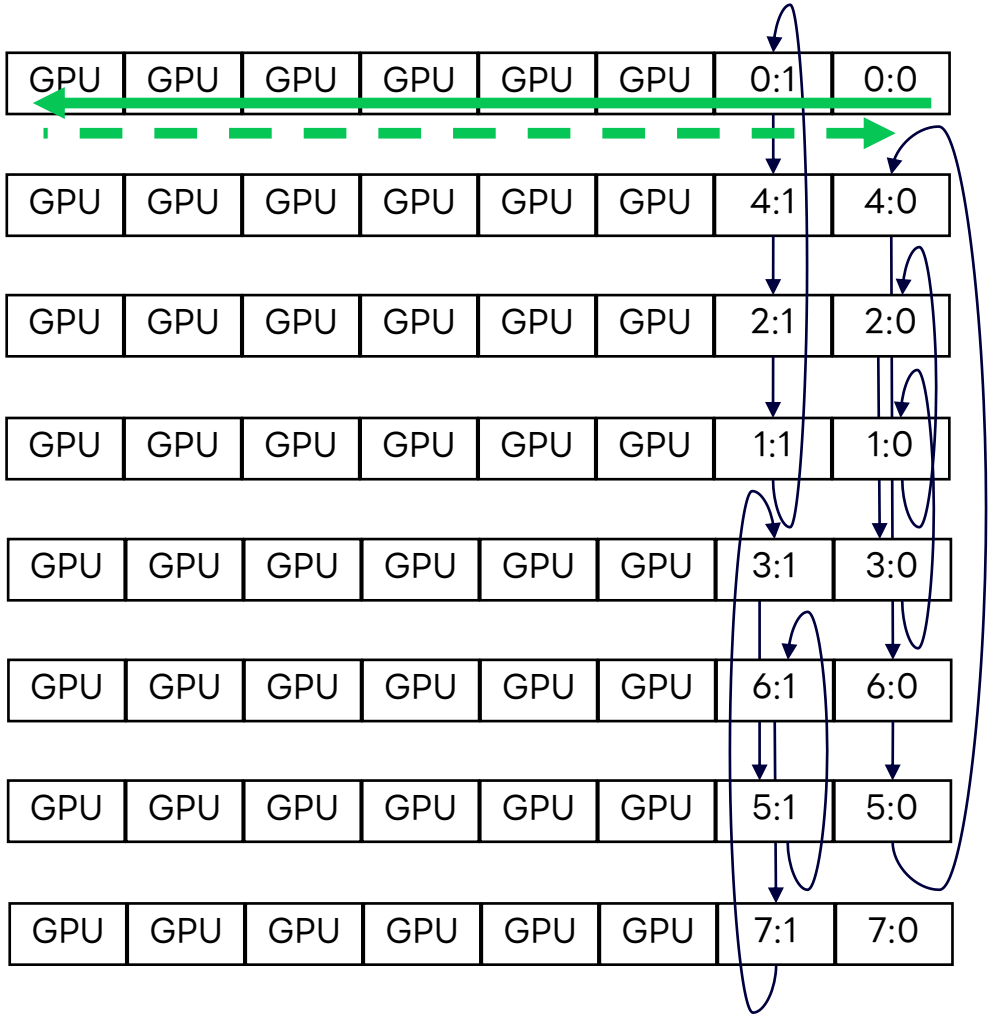
偏りなくRailの帯域を飽和させるアルゴリズム

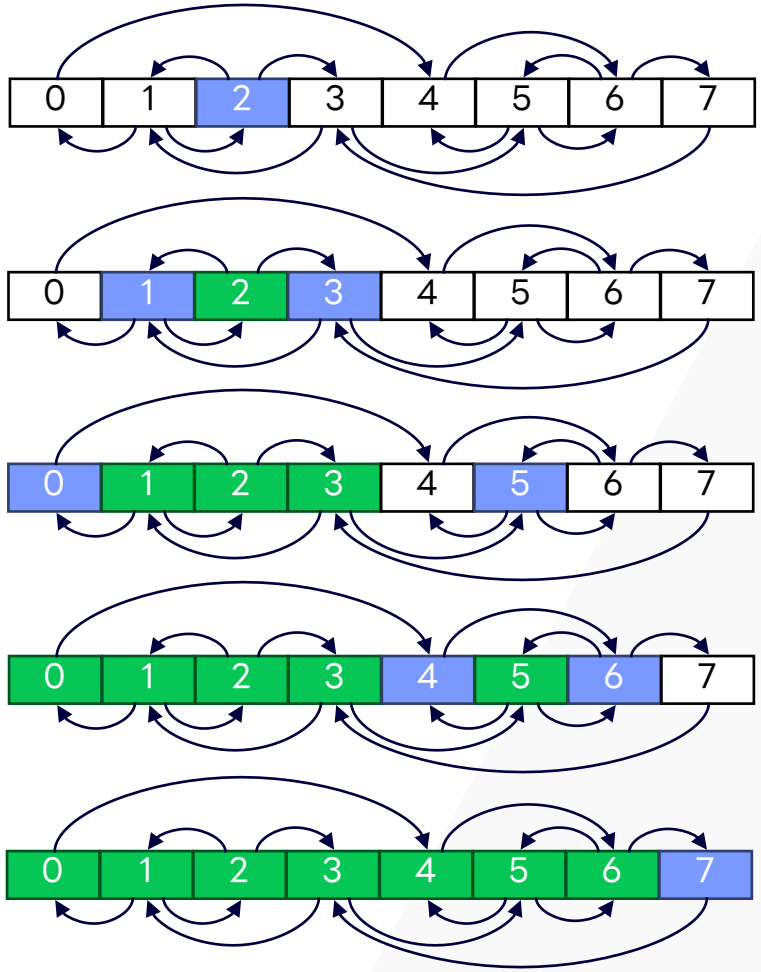




Balanced Tree Algorithm

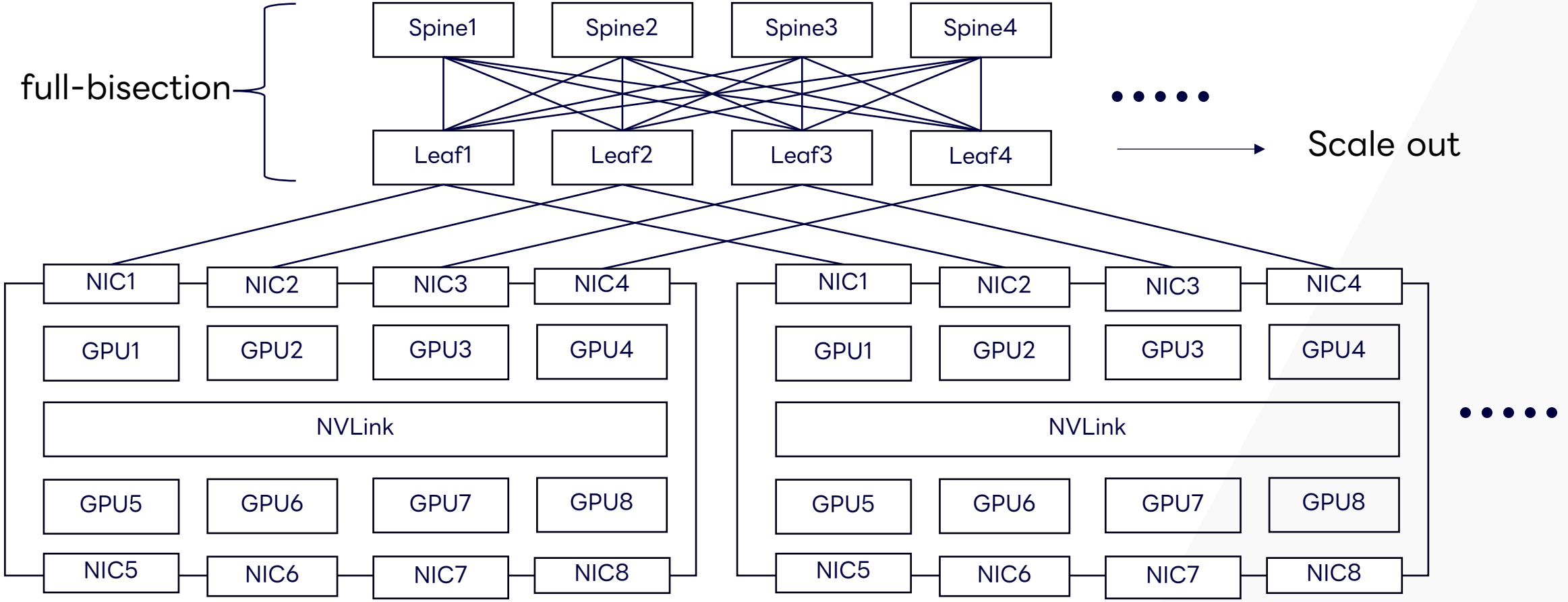
一部リンクを犠牲にStep数を減らすアルゴリズム





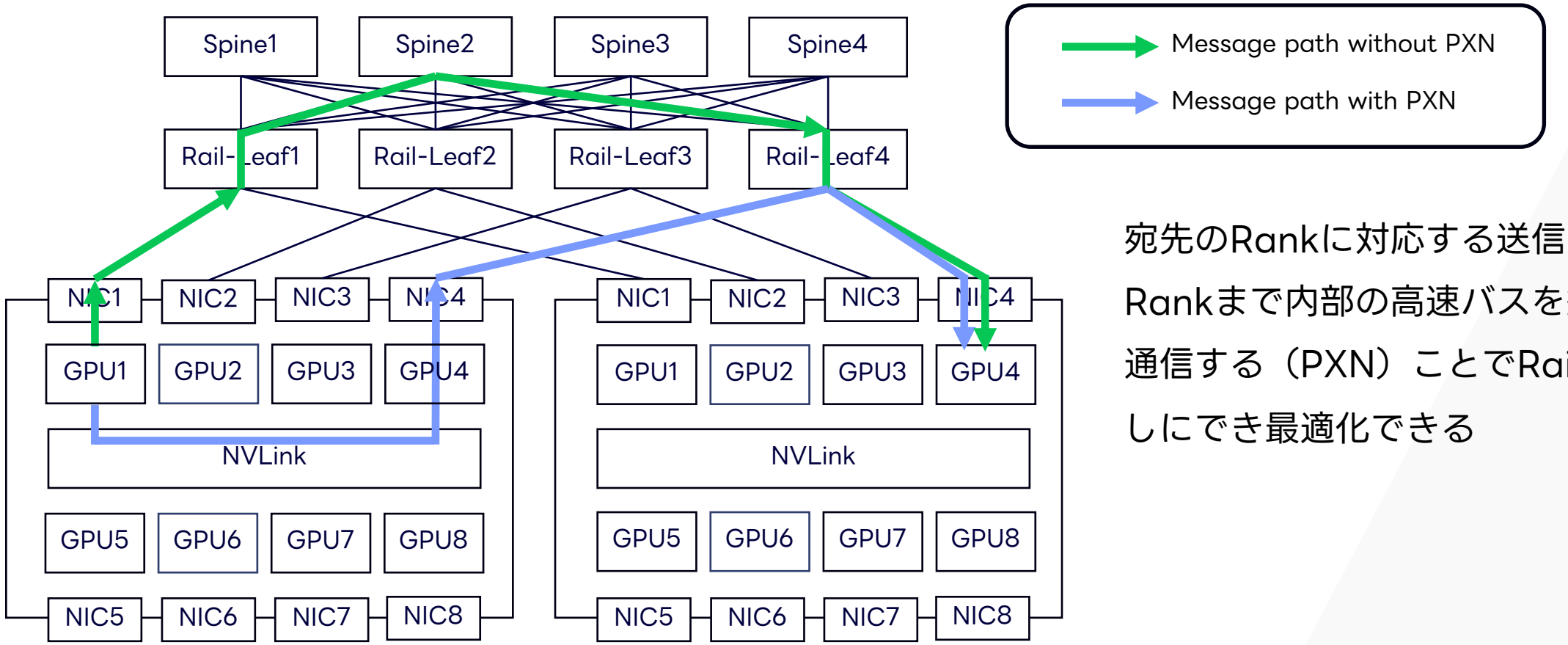
Rail-Optimized Topology

集合通信の性能を最適化しつつ、スケールアウト可能にする構成



PXN（PCIe x NVLink）とは

内部の高速バスを経路として利用する技術

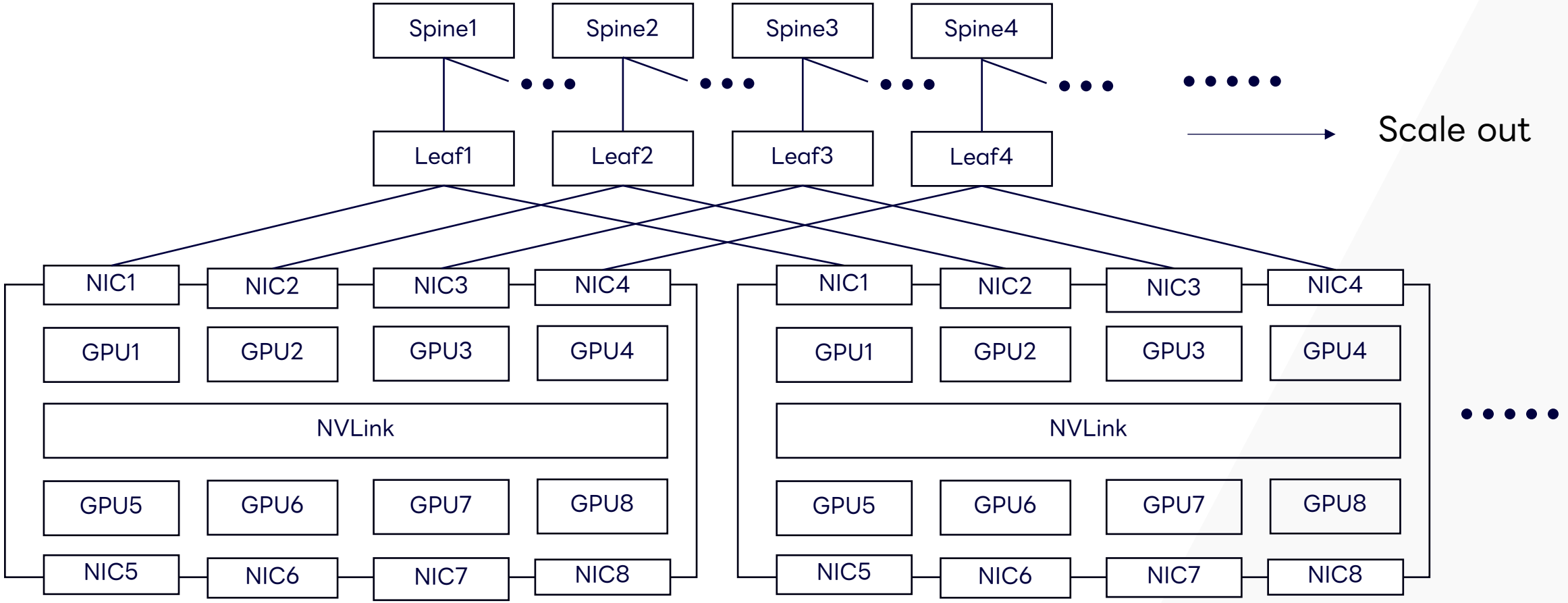


宛先のRankに対応する送信元のRankまで内部の高速バスを経由し通信する（PXN）ことでRail折り返しにでき最適化できる

Ref: https://developer.nvidia.com/blog/doubling-all2all-performance-with-nvidia-collective-communication-library-2a-12/

Rail-Only Topology

集合通信の性能を最適化しつつ、スケールアウト可能にする構成



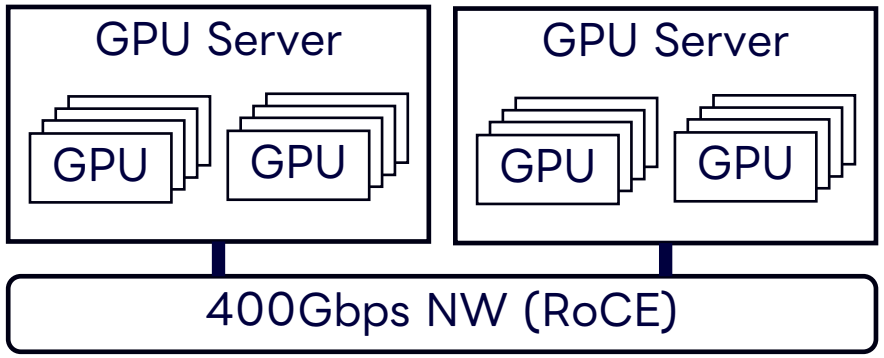
どのTopologyを選択するか？

多角的な視点からのNWトポロジーの選択

	Single Chassis	Rail-Optimized	Rail-Only
スケール	×	○	◎
コスト	○	△	○
運用難度	○	△	？
運用実績	×	○ （CLOSの運用実績が豊富）	？

サーバ構成・サーバ内部NW構成

議論のスコープ

	分散学習向けNW
課題・要求事項	
技術的要求 考慮事項	• GPU NWの構成 • 仮想化の廃止
我々の選択	• GPU NWの構築 • サーバ構成・サーバ内部NW構成

以降の議論の流れ

- GPUサーバの選択
- GPUサーバ自体のアーキテクチャ
- サーバ内部NWのアーキテクチャ

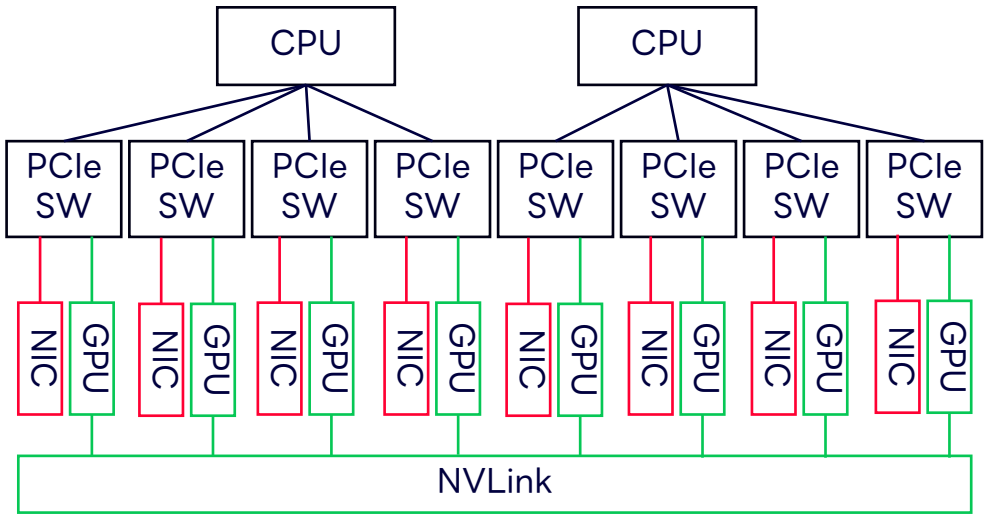
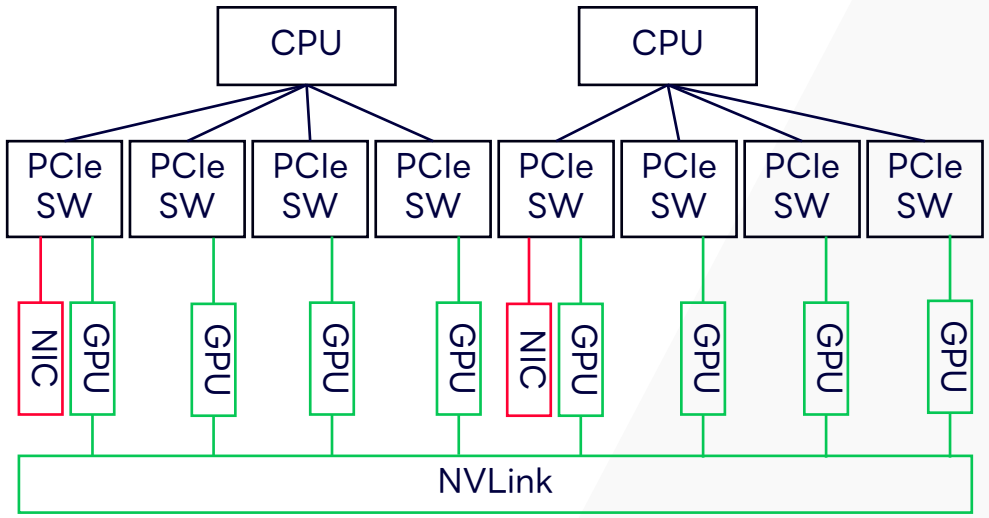
GPUサーバの選択

GPU搭載サーバの違いと我々からみた違い

	PCIe	HGX	DGX
OS/SW 自由度	○	○	×
内部バス 性能	△	○	○
流通速度	×	△	○
サポート	×	×	○
URL	https://www.nvidia.com/ja-jp/data-center/h100/	https://www.nvidia.com/ja-jp/data-center/hgx/	https://www.nvidia.com/ja-jp/data-center/dgx-h100/

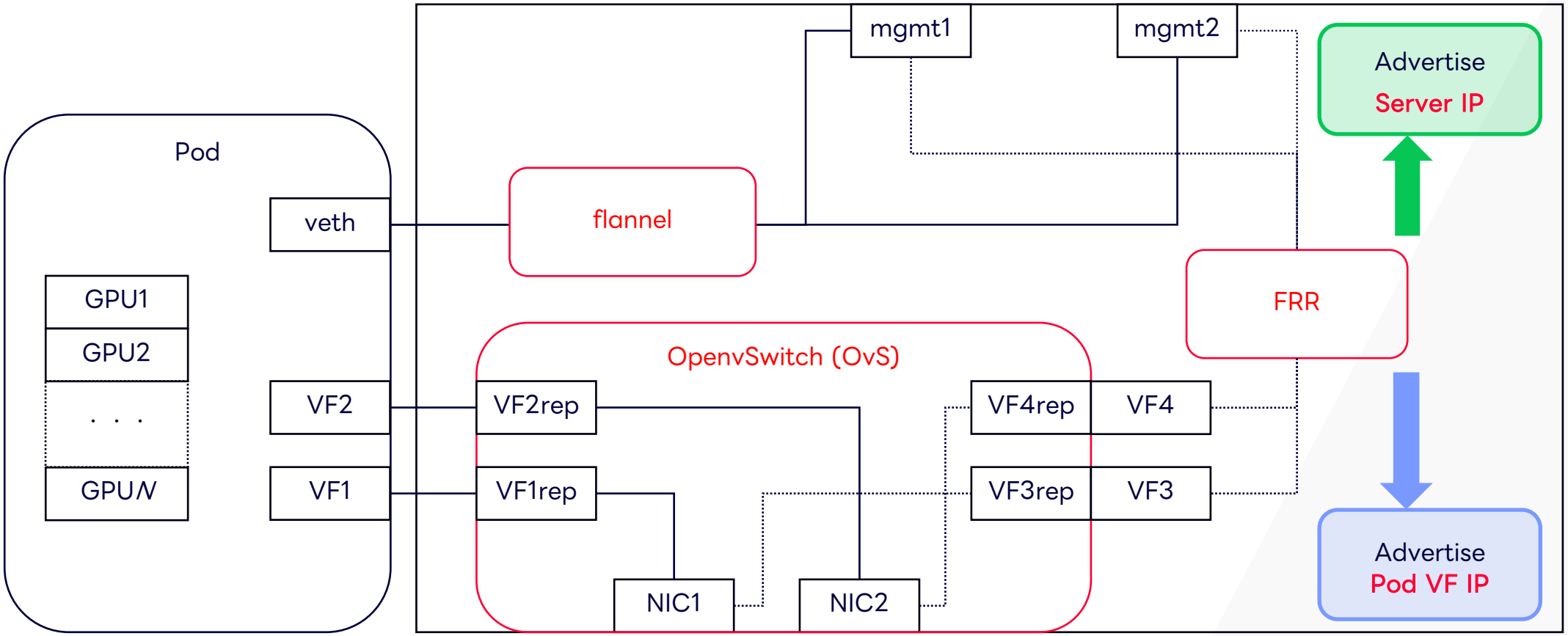
Server Architecture

HGX H100 サーバ構成

	Reference Architecture	ACP H100 Architecture
Diagram		
特徴	NWの帯域性能が一番出るがコストは高い GPUx8に対しそれぞれNICがある形 Rail-LeafもNICに合わせて8個必要	NICの枚数を減らしRailを削減する構成 既存ワークロード負荷を踏まえた最小構成 コストは低いが性能が悪い（詳細後述）

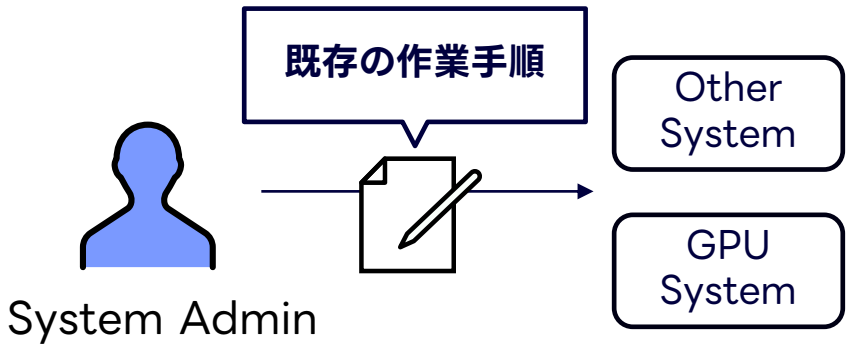
Server Network Architecture

従来の運用通りに保ちつつ、柔軟で最速なNWを実現する



BGPベースの構成による安定運用

議論のスコープ

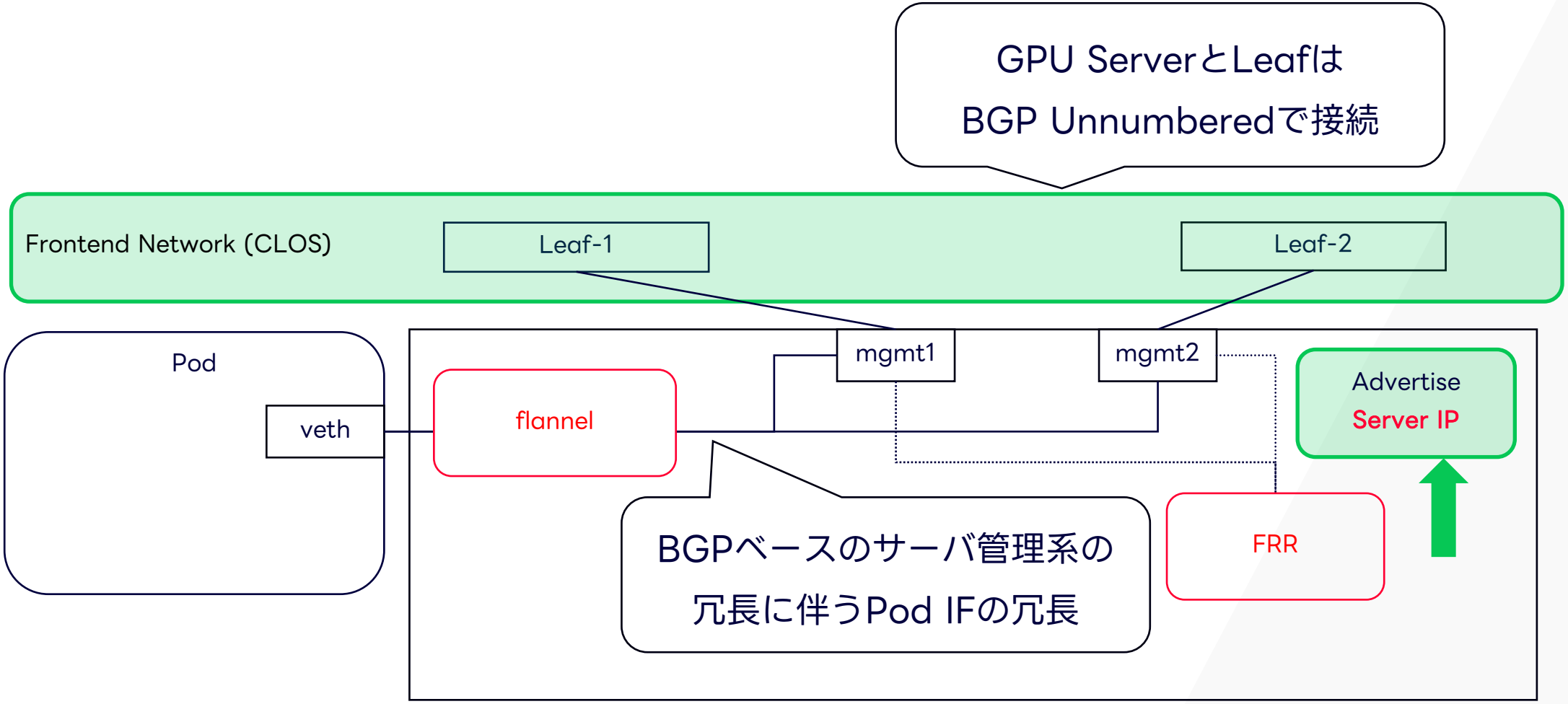
	運用互換性・既存資産の流用
課題・要求事項	
技術的要求 考慮事項	• 構成差分の最小化 • 管理NWの冗長
我々の選択	• BGPベースのNW構成 • BGPベースの管理NWの冗長

以降の議論の流れ

- サーバ管理系通信
  （Frontend Network）
  のためのサーバ内NW詳細

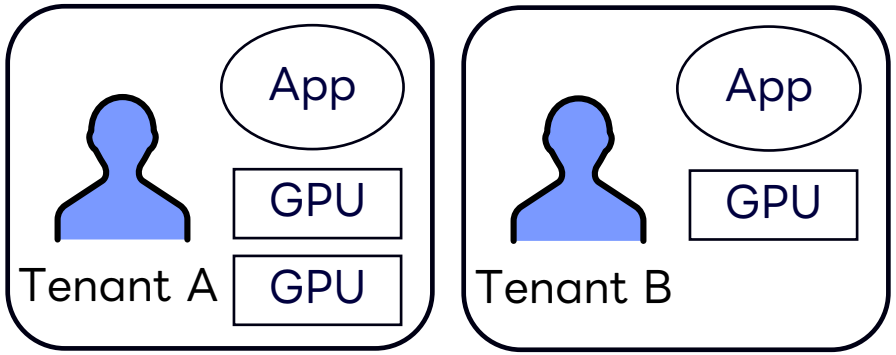
Frontend Network Architecture

サーバ管理系、Podの管理系（表）通信と安定運用のための互換性



GPU NW構成の詳細とテナント分離の実現策

具体的な要求と選択したソリューション

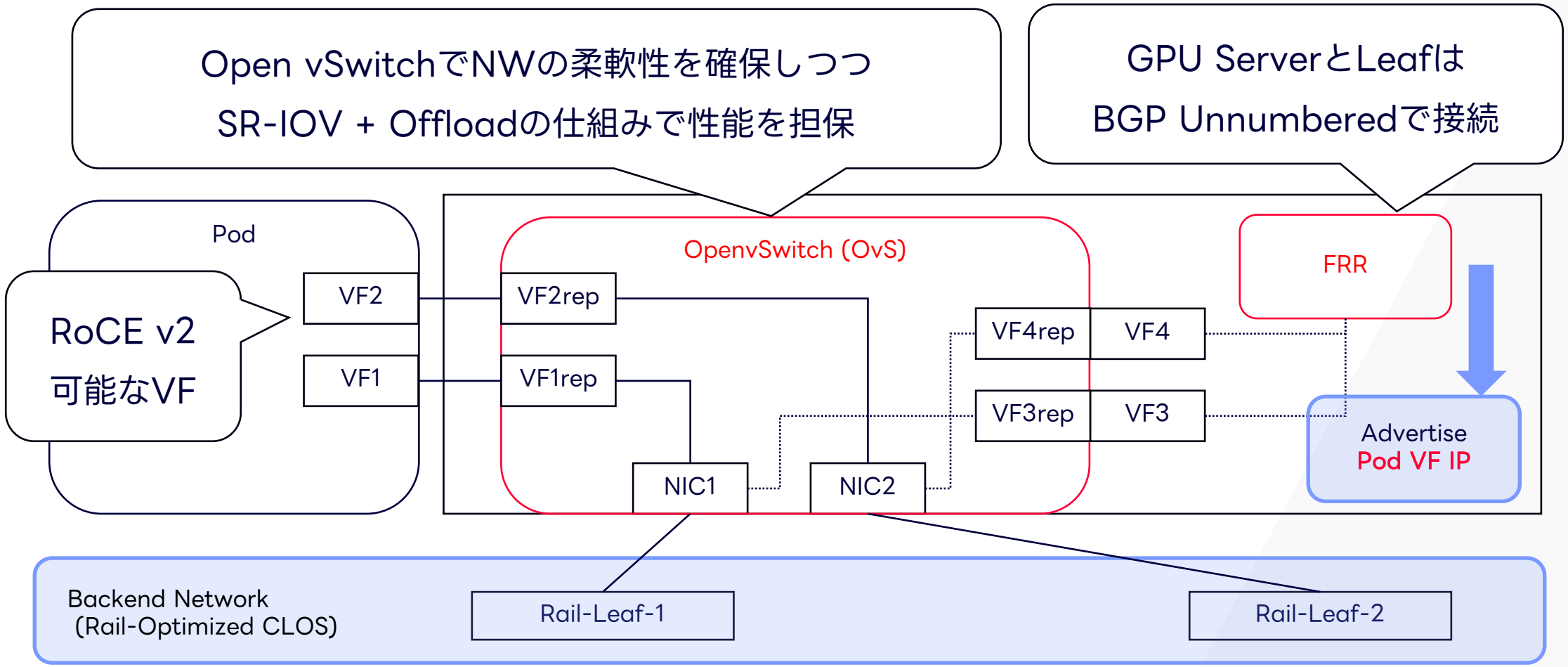
	Multi-Tenancy
課題・要求事項	
技術的要求 考慮事項	• プラットフォーム上のテナント分離 • パフォーマンス影響のないNW分離
我々の選択	• k8s namespaceによるテナント分離 • OpenvSwitchによるNW分離

以降の議論の流れ

- GPU NW（Backend Network）のためのサーバ内NW詳細
- OvS Hardware Offload
- プラットフォーム（k8s）との連携とテナント分離の実現策

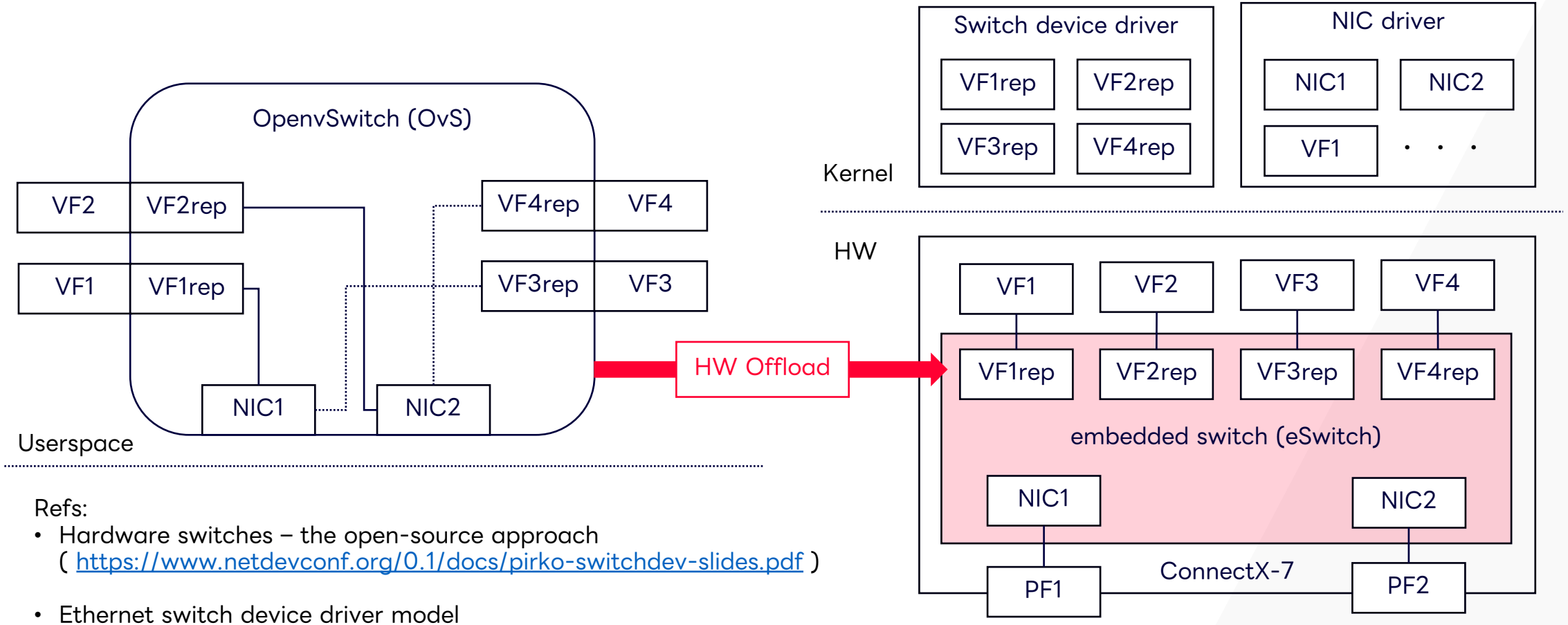
Backend Network Architecture

GPU向け通信とNWの柔軟性の確保



OvS Hardware Offload

性能を担保したまま、OvSによる柔軟性を獲得

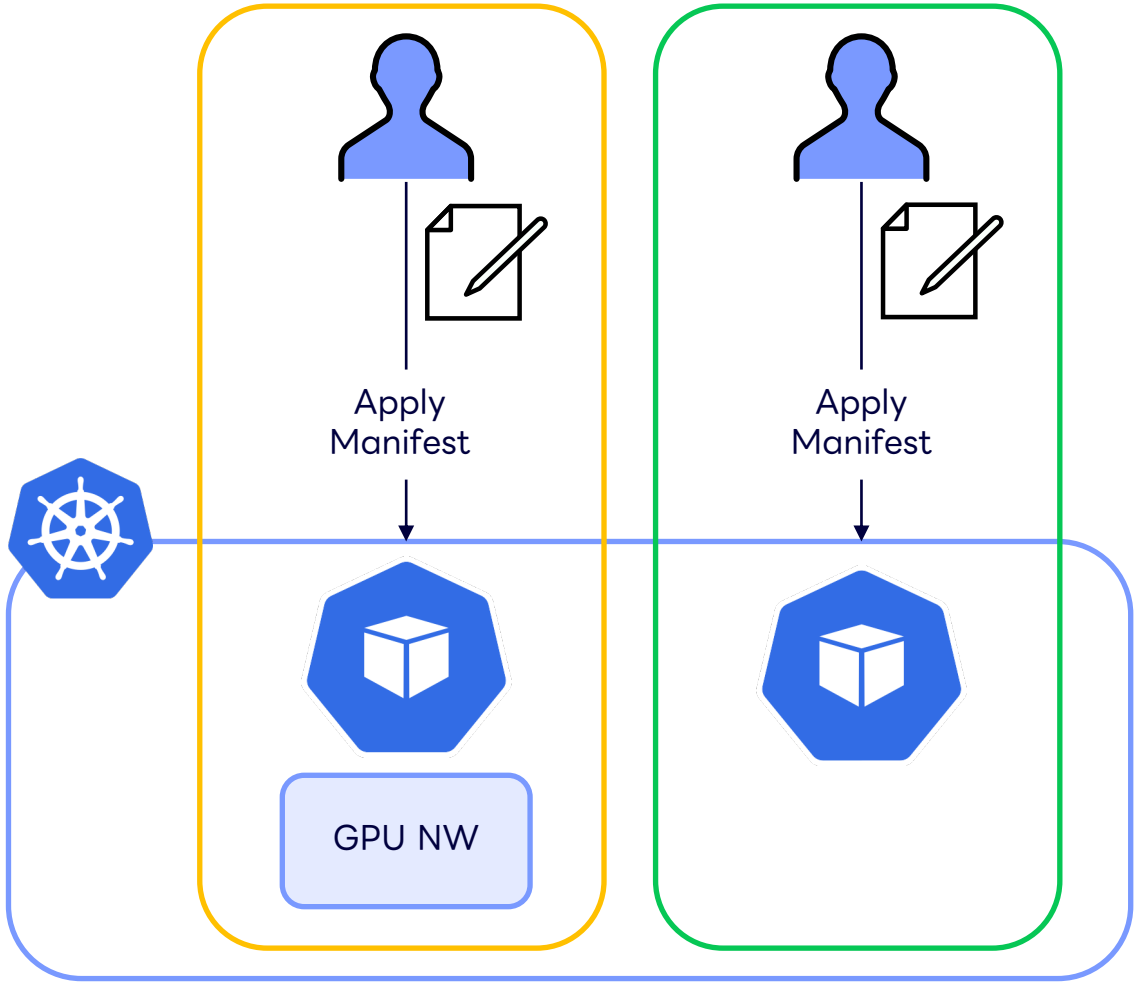


Refs:

- Hardware switches – the open-source approach
  (https://www.netdevconf.org/0.1/docs/pirko-switchdev-slides.pdf)
- Ethernet switch device driver model
  (https://docs.kernel.org/networking/switchdev.html)

Kubernetesとの連携

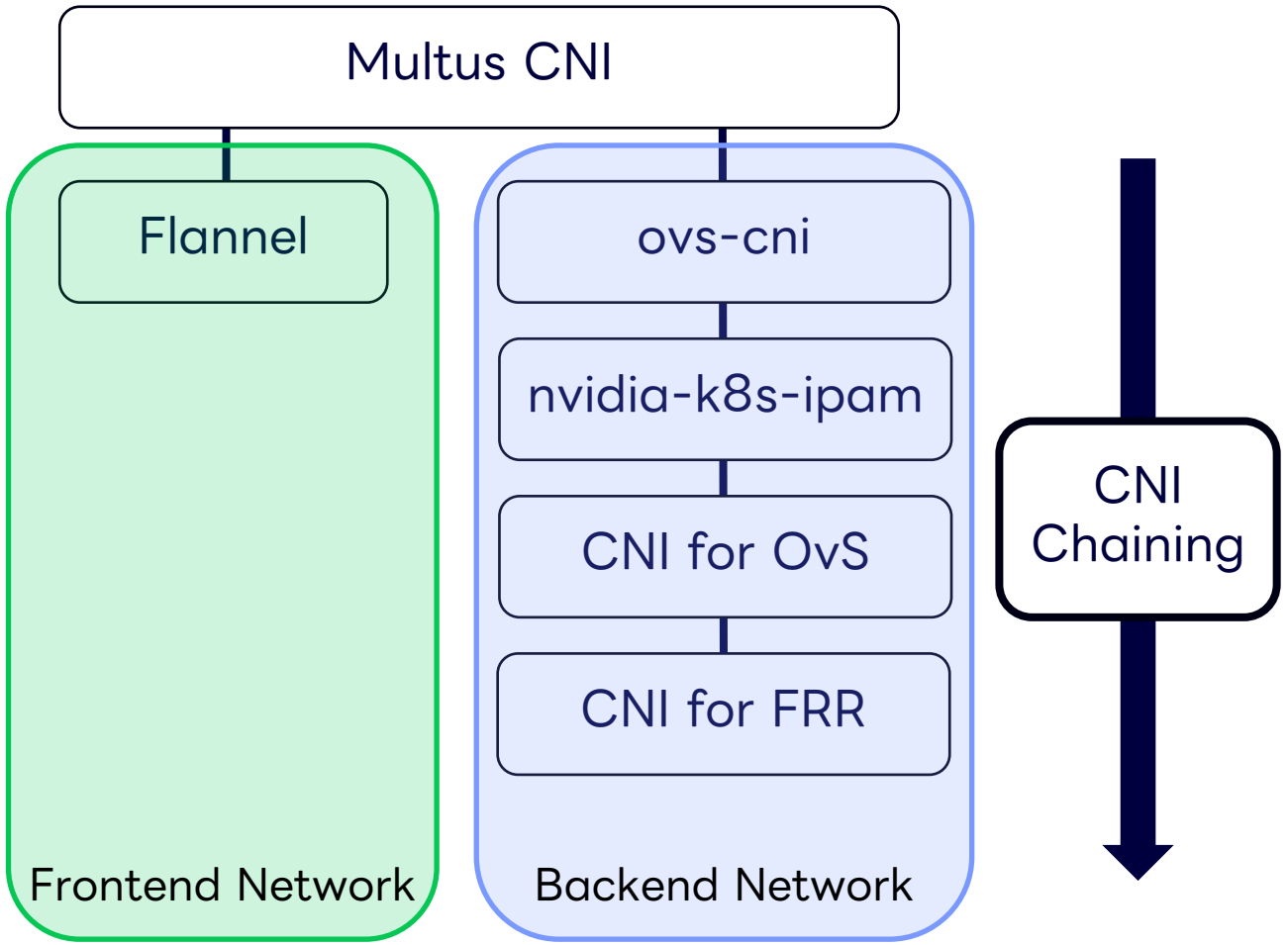
KubernetesベースのPlatform（ACP）とサーバ内NW Solutionの統合



- Pod起動時にユーザが追加NICを選択した場合
  そのPodにGPU NWを敷設したい
    - 逆にGPU NWが不要な場合は使わない
- Pod起動時にGPU NWを敷設するのはCNIの役割
- 既存のソリューションで実現できるか？
  → 要求を全て満たすものは無いので内製を検討

内製の方針

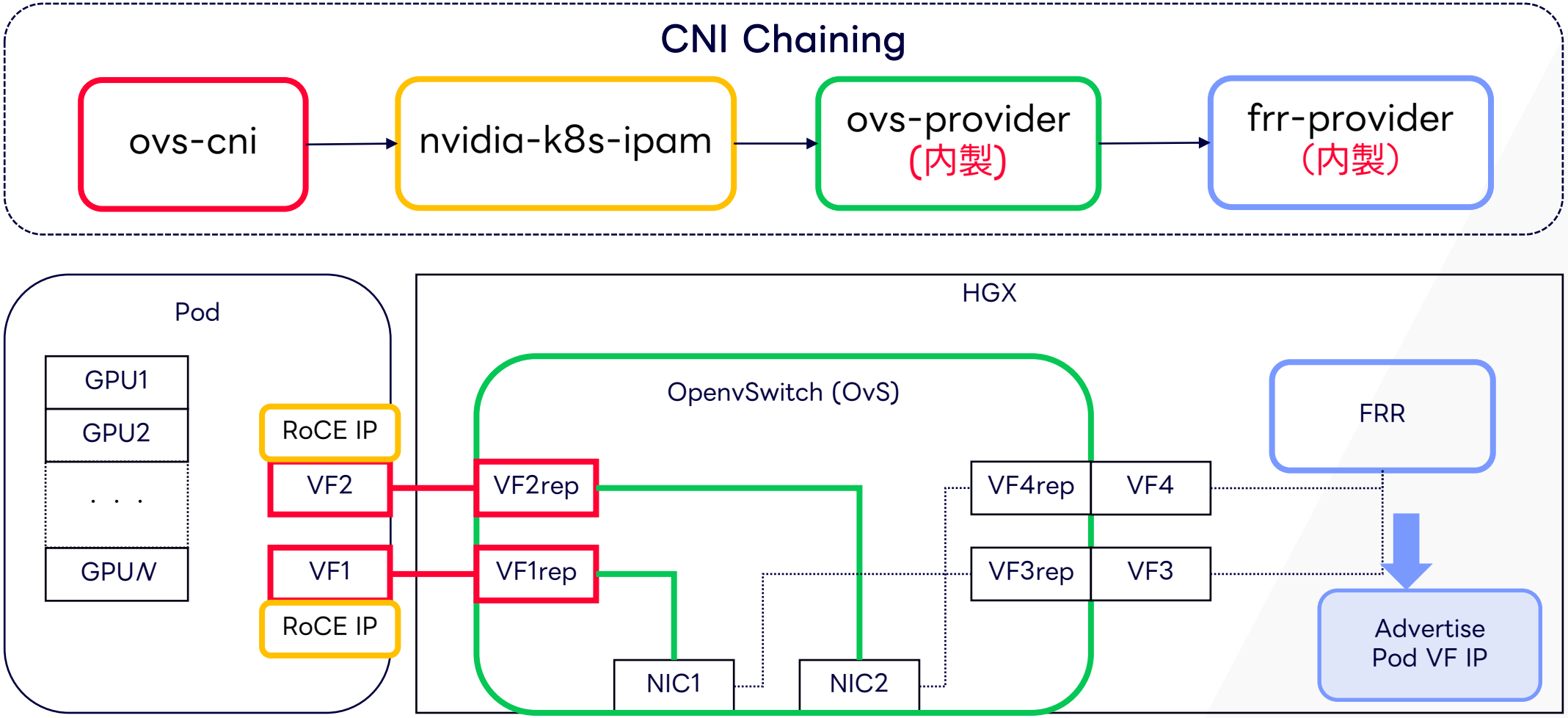
どのように足りない機能を実現するか？



- Multus CNIの利用とPodへの複数IF付与
- CNIのSpecにはChainの仕組みがあるため、裏NWのCNIではこの仕組みをベースに構成
- 必要な機能毎にPluginをChainさせる方式を採用
  - インターフェイスの設定はovs-cniを利用
  - IPAMはnvidia-k8s-ipamを利用
  - OvS/FRRの設定をするPluginを自作

Kubernetesとの統合

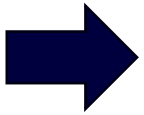
CNIの全体の流れとChaining



利用者からみた変更

Kubernetes manifestの変更点

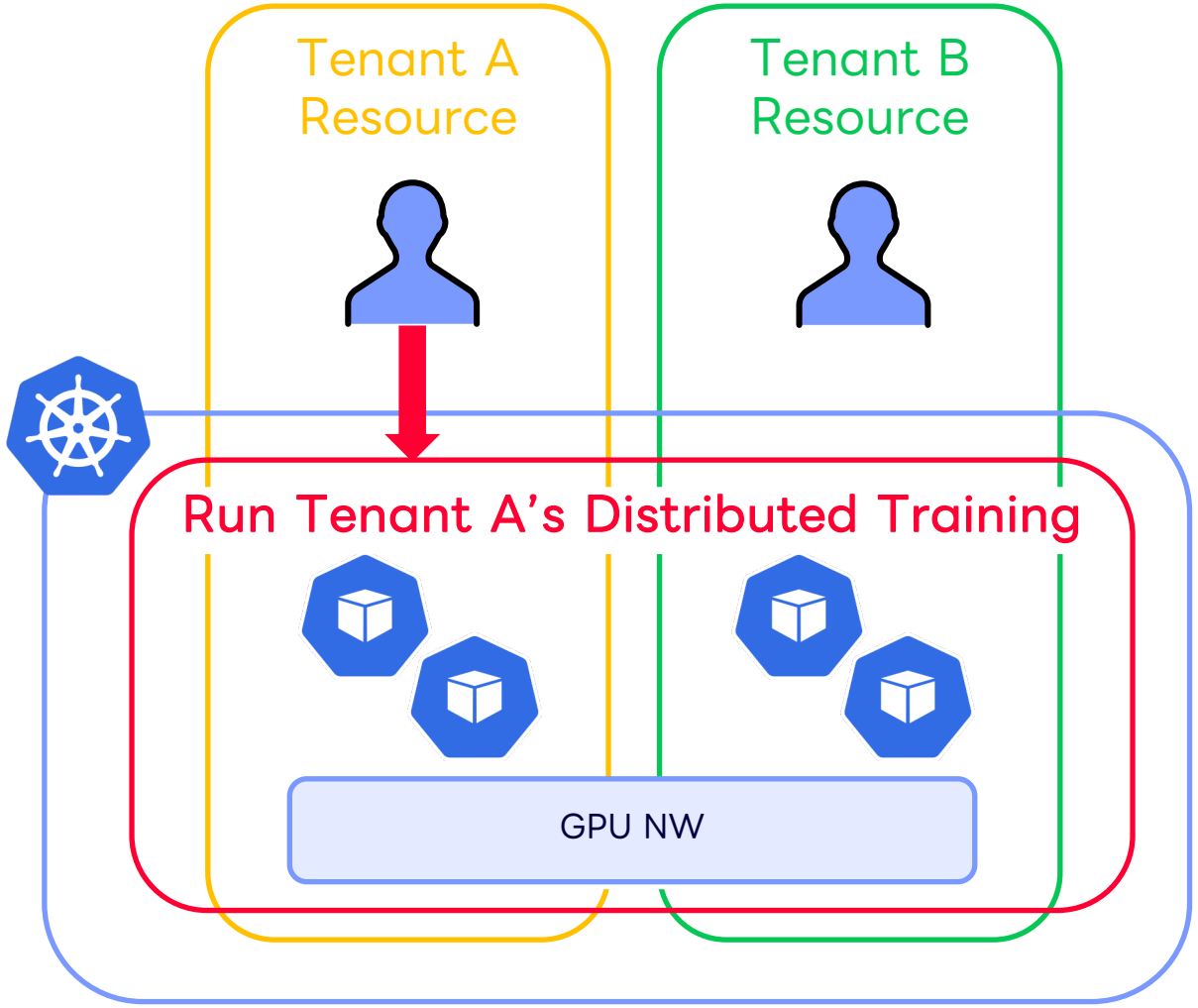
```
---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
  namespace: tenant1
spec:
  …
  containers:
    …
    resources:
      requests:
        nvidia.com/gpu: 8
    …
```



```
---
apiVersion: v1
kind: Pod
metadata:
  name: gpu-pod
  namespace: tenant1
  annotations:
    k8s.v1.cni.cncf.io/networks: sriov_pf1_vf,sriov_pf2_vf
spec:
  …
  containers:
    …
    resources:
      requests:
        nvidia.com/gpu: 8
        nvidia.com/mlnx_sriov_pf1_vf: 1
        nvidia.com/mlnx_sriov_pf1_vf: 1
    …
```

ACP Tenant Isolation

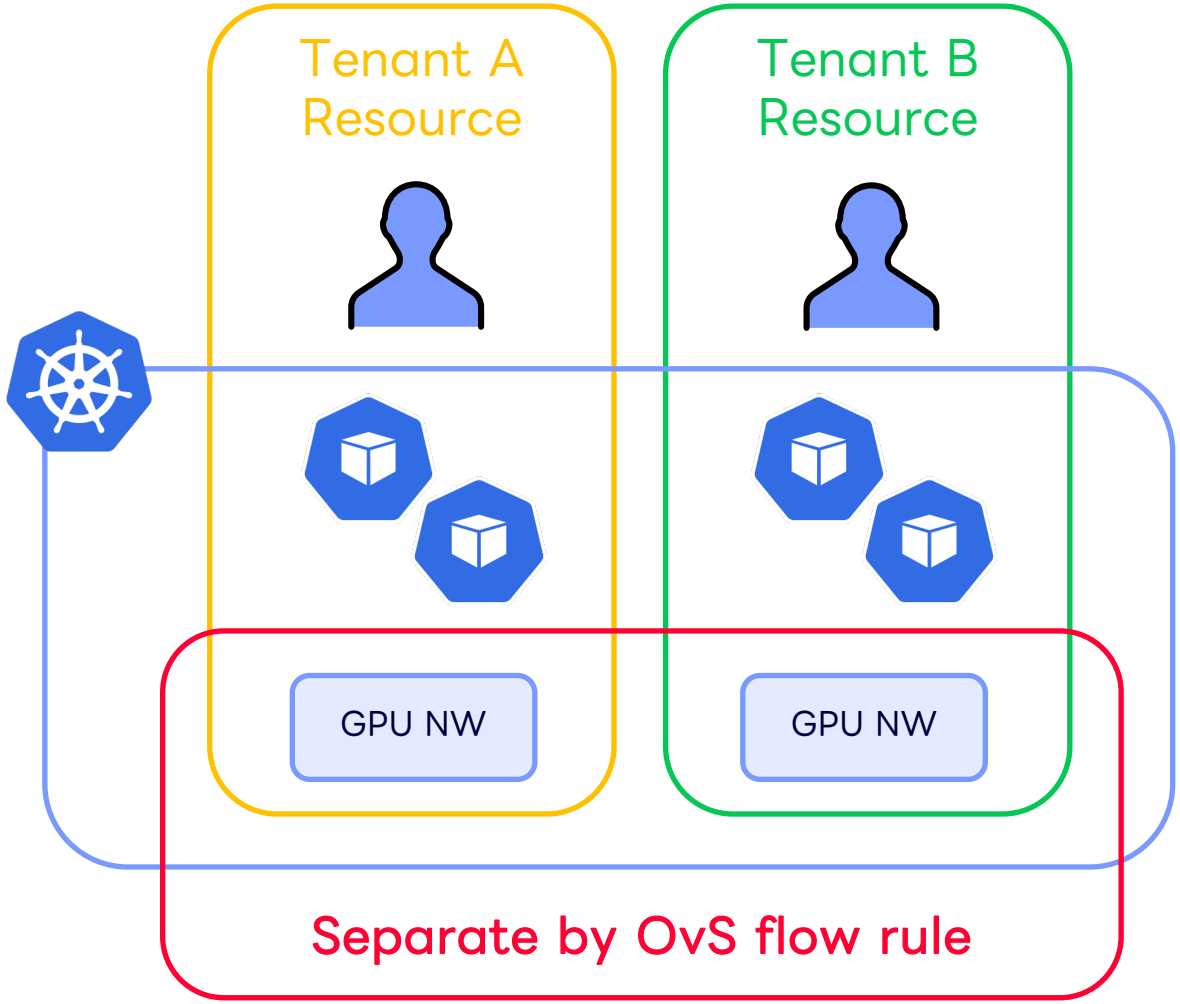
ACPにおけるテナント分離の方式とGPU NWにおけるテナント間分離の必要性



- ACPはNamespaceベースのテナント分離
- NWを適切に分離しないと、条件次第では別テナントのGPU Podを巻き込んで分散学習を実行できる
- 管理NW（Podの通常IF側）はNetwork Policy
- GPU NWを適切に分離する必要がある

GPU NW Tenant Isolation

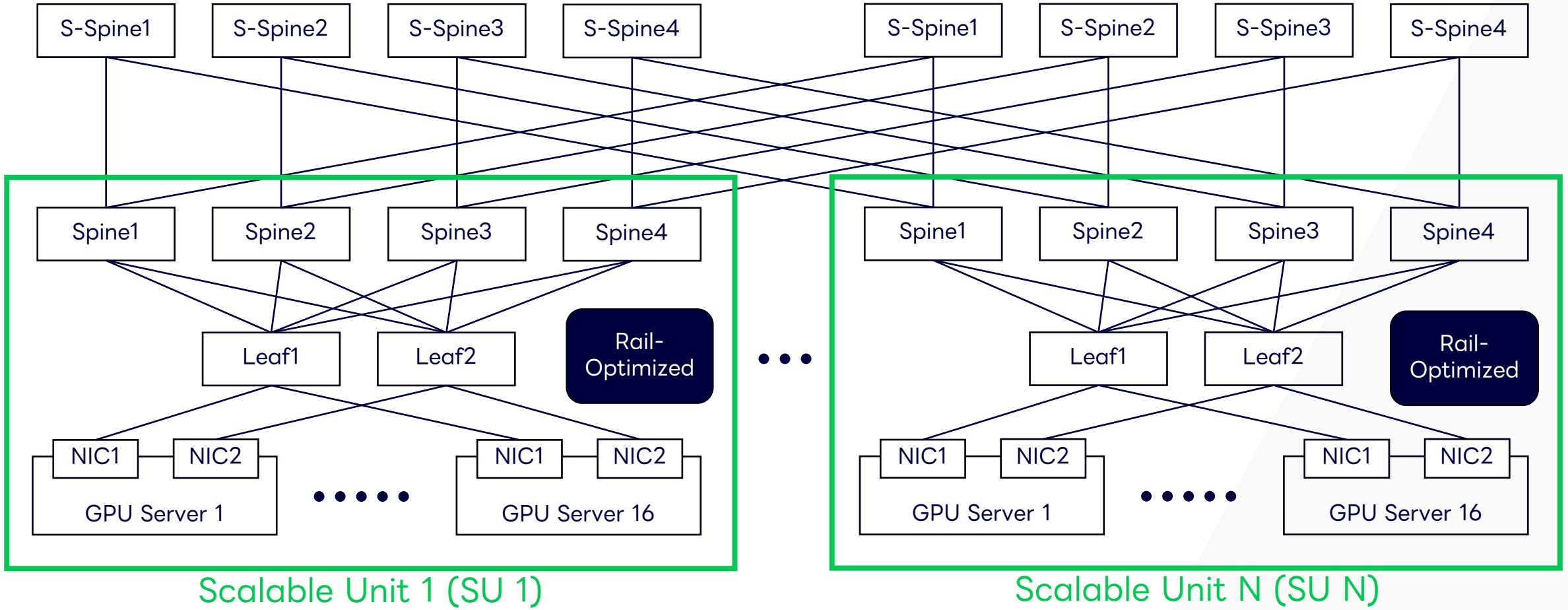
CNI + OvS + HW Offloadを利用したGPU NWにおけるテナント間分離方式案



- GPU NWの分離方式の一つとして、OvSのフロールールベースでの調整が考えられる
- CNIリクエストにはNamespaceが含まれるため、CNIの形でOvSを設定する仕組みが、Namespaceベースのテナント分離方式と相性が良い。
- GPU NWは性能が命なので、これはOvSのOffloadができることが前提の仕組み

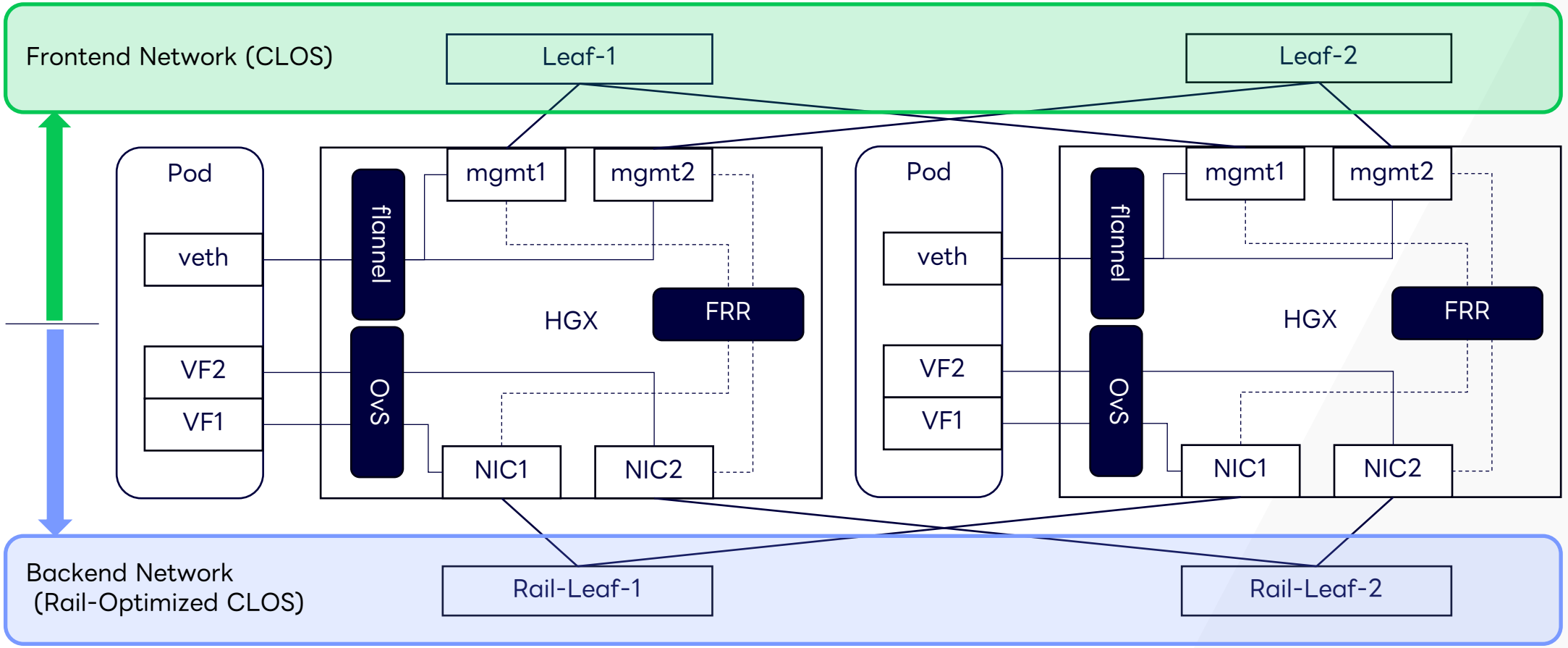
GPU NWの全体図

採用した技術と最終的な構成



サーバ構成・サーバ内NWの全体図

採用した技術と最終的な構成

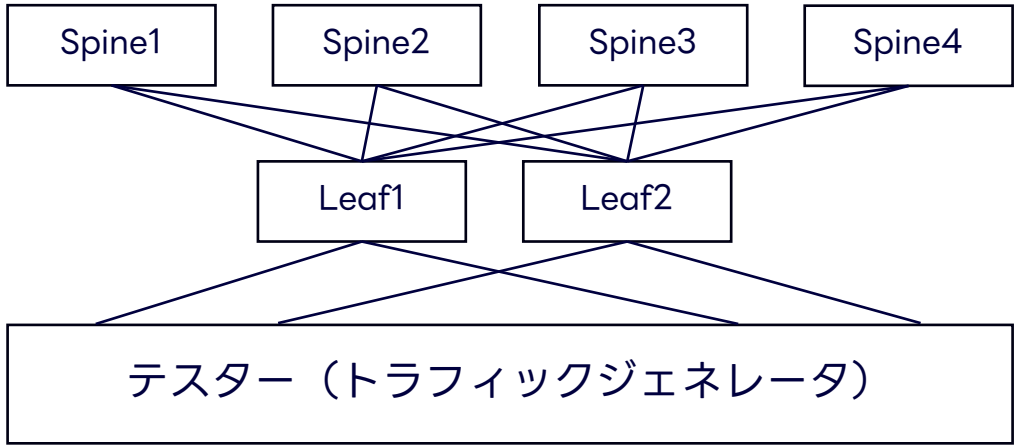


実際に設計・構築して
どうだったか？

GPU NWの遅延計測

400Gbps RoCEv2対応テスターによる測定

測定時の構成



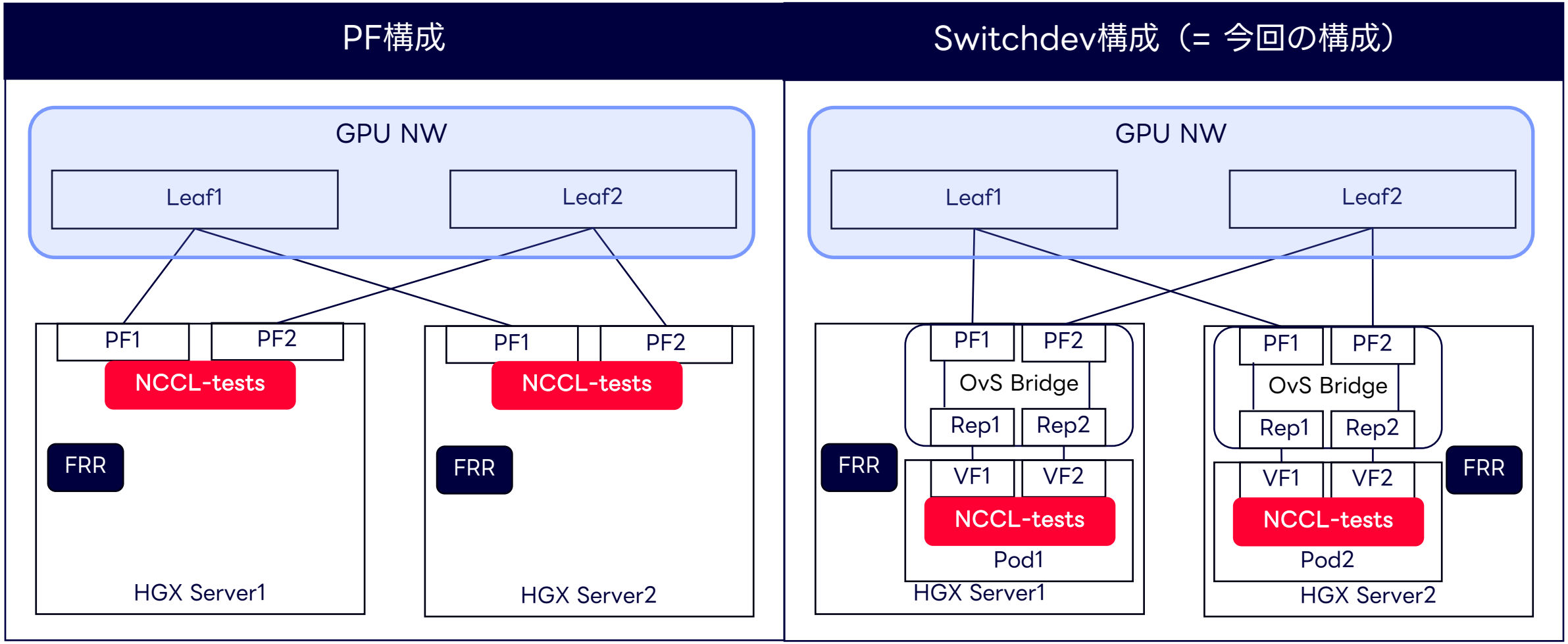
写真

測定結果

	Leaf折り返し(us)	Spine折り返し(us)
Avg Latency	1.13	3.27
Min Latency	1.11	2.98
Max Latency	1.39	4.20

性能試験の構成

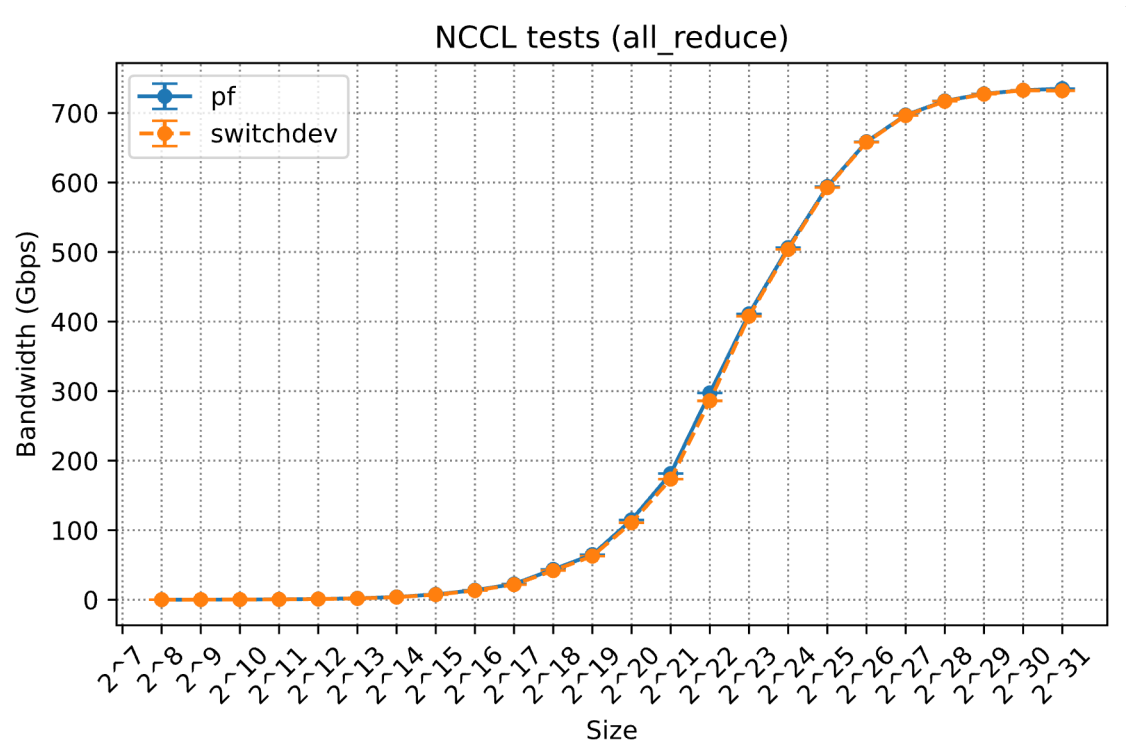
検証環境



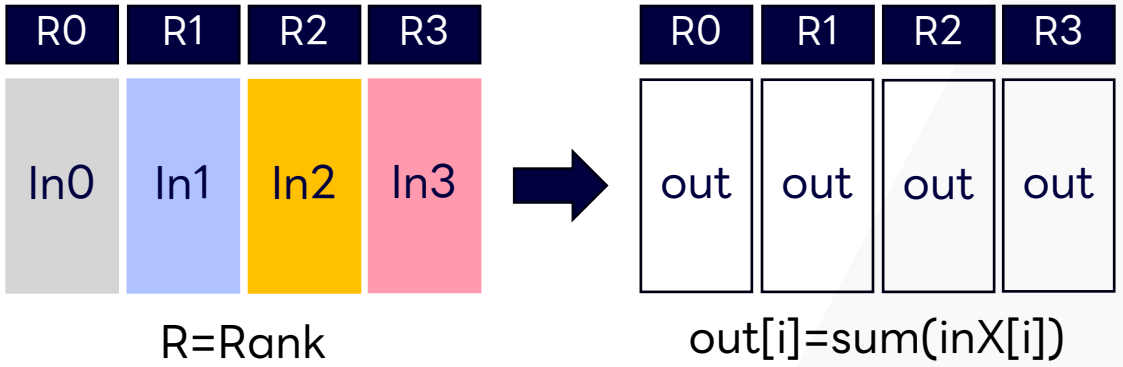
NCCL-testsを利用した測定

All-Reduceの性能測定結果

性能測定結果 (Node=2)



通信イメージ



R=Rank

out[i]=sum(inX[i])

特性・用途

全Rankのデータを Reductionし共有する計算

分散学習における勾配の計算・分配に

最も良く利用される

Ref: https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/usage/collectives.html

理想と現実のギャップ

うまく行ったと思いきや…想定外の出来事も

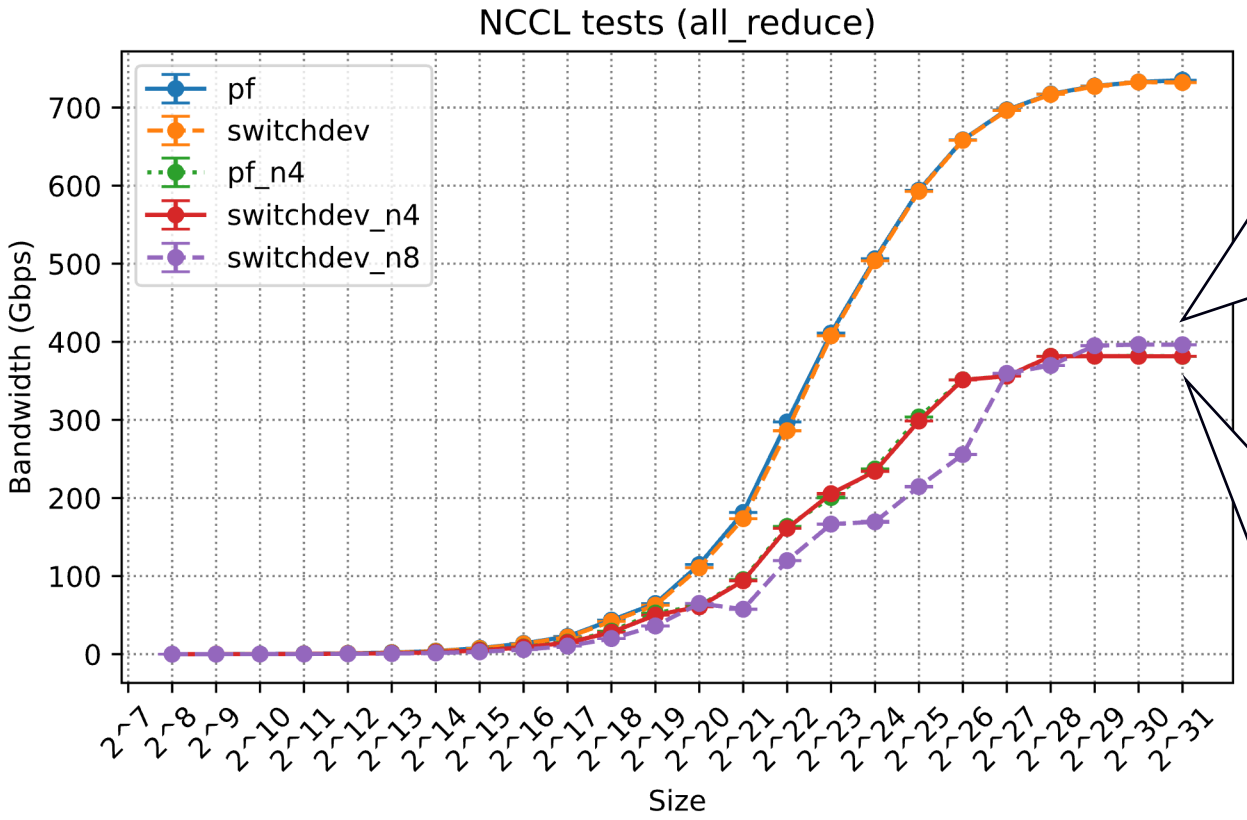
4 Server以上での性能傾向

PXNの認識の誤解

Server数を増やした場合の性能変化

All-Reduceの帯域性能が悪くなる

AllReduce 性能測定結果



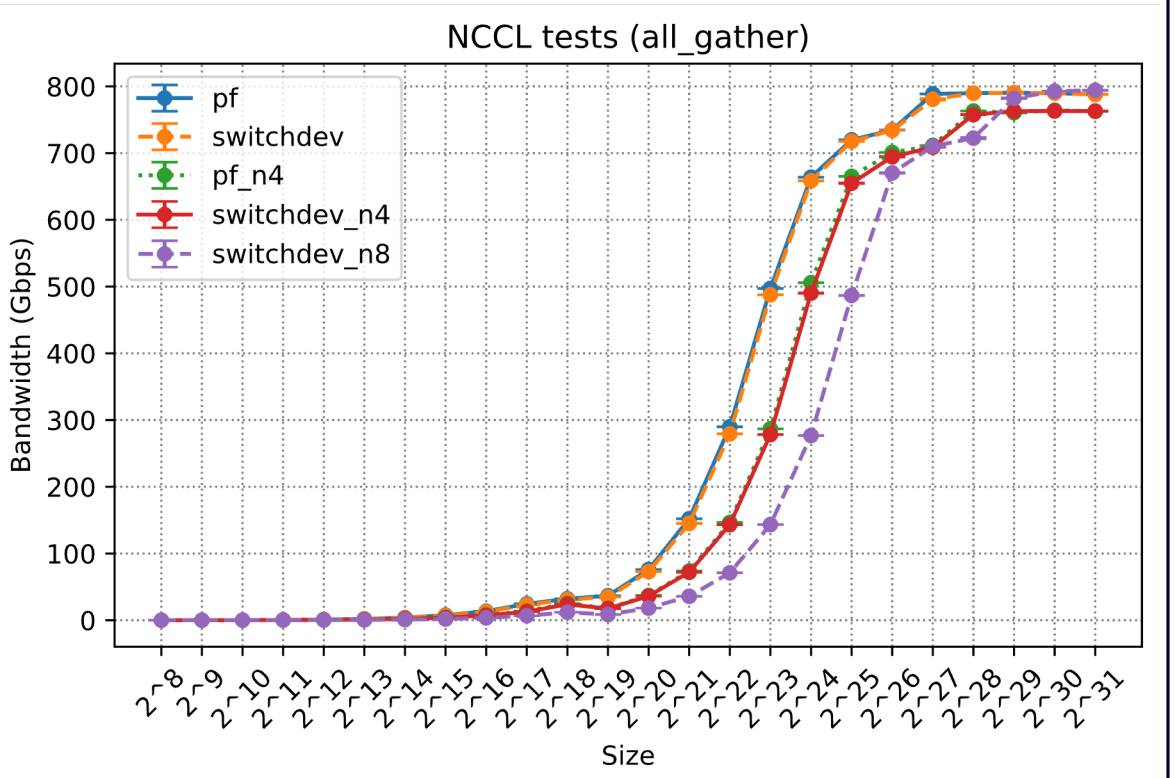
PF構成もSwitchdev構成も
Node=4を超えると
最大帯域が400Gbpsで律速？

Server台数を4から8に増やしても
同様の傾向

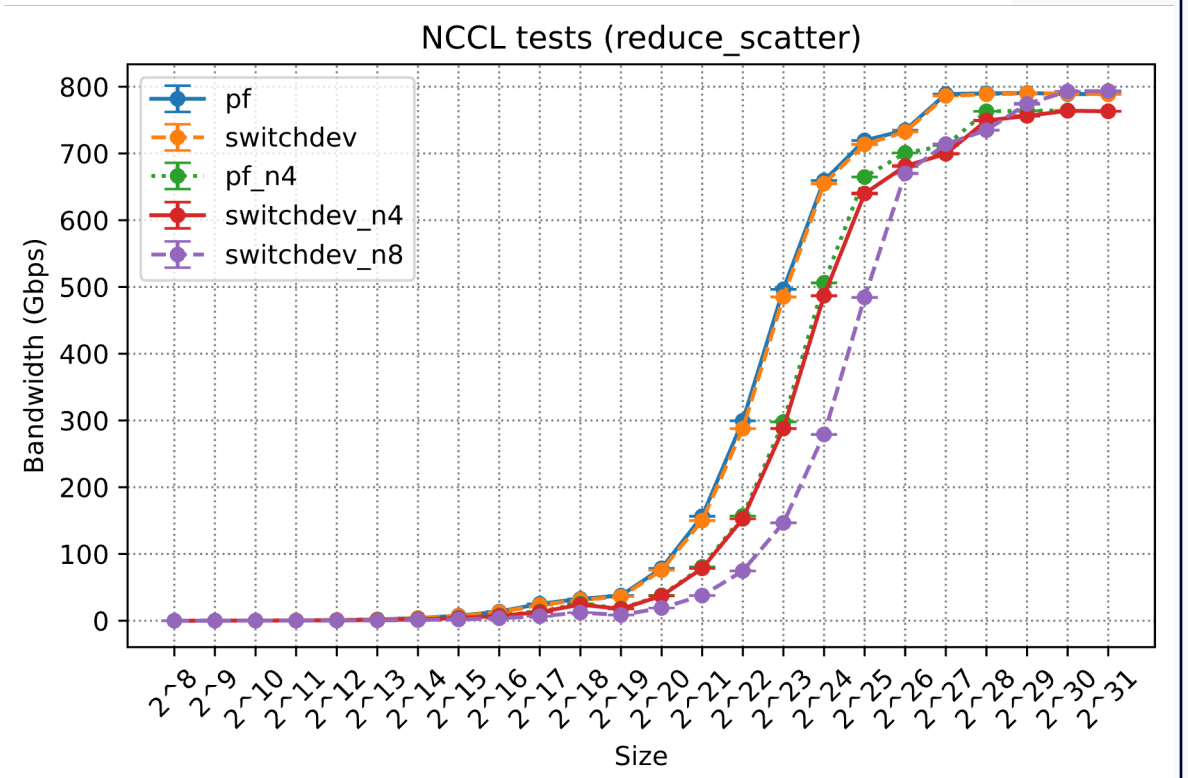
Server数を増やした場合の性能変化

All-Gather, Reduce-Scatterは期待した帯域性能が出ている

AllGather 性能測定結果



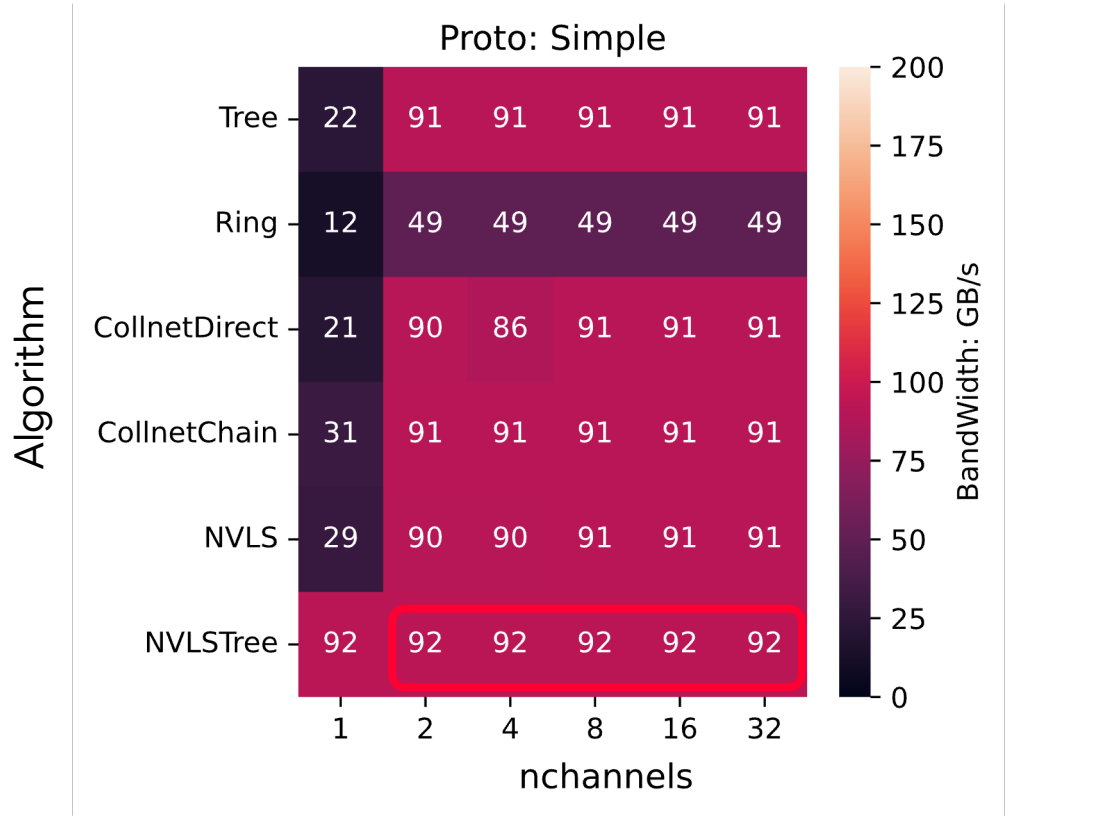
ReduceScatter 性能測定結果



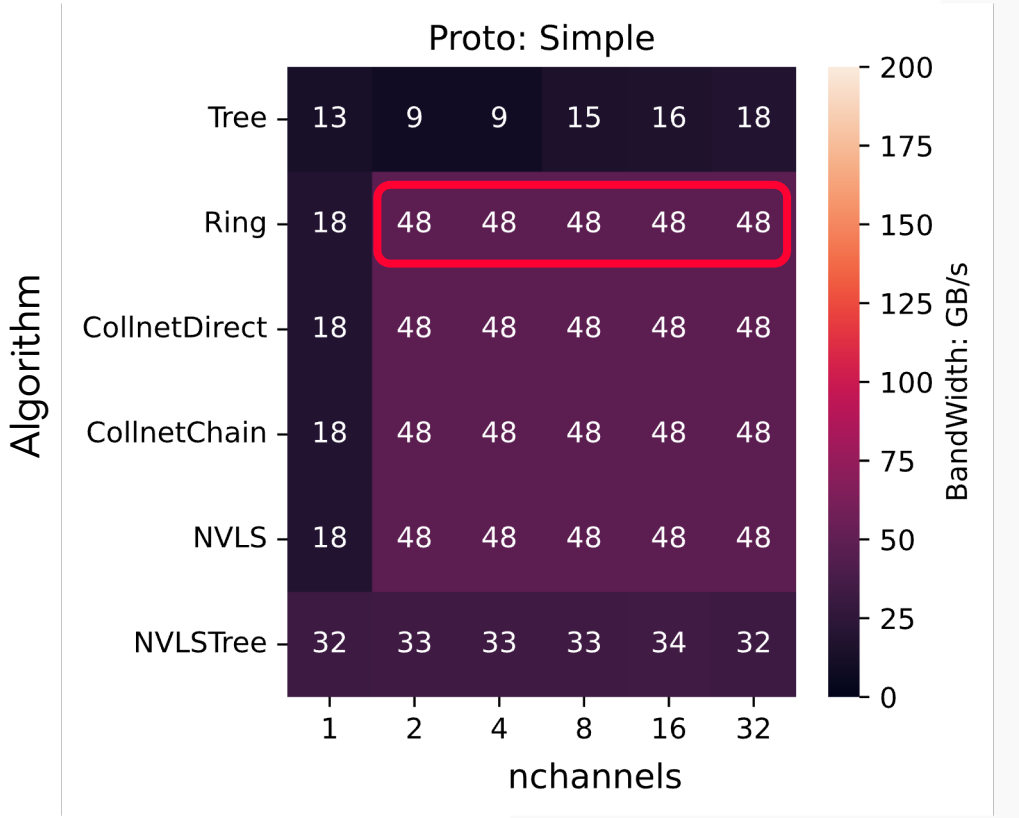
変量を調整した場合の性能変化

Nodeの台数の変化で自動選択されるアルゴリズムが変わった

Node=2、NIC=2



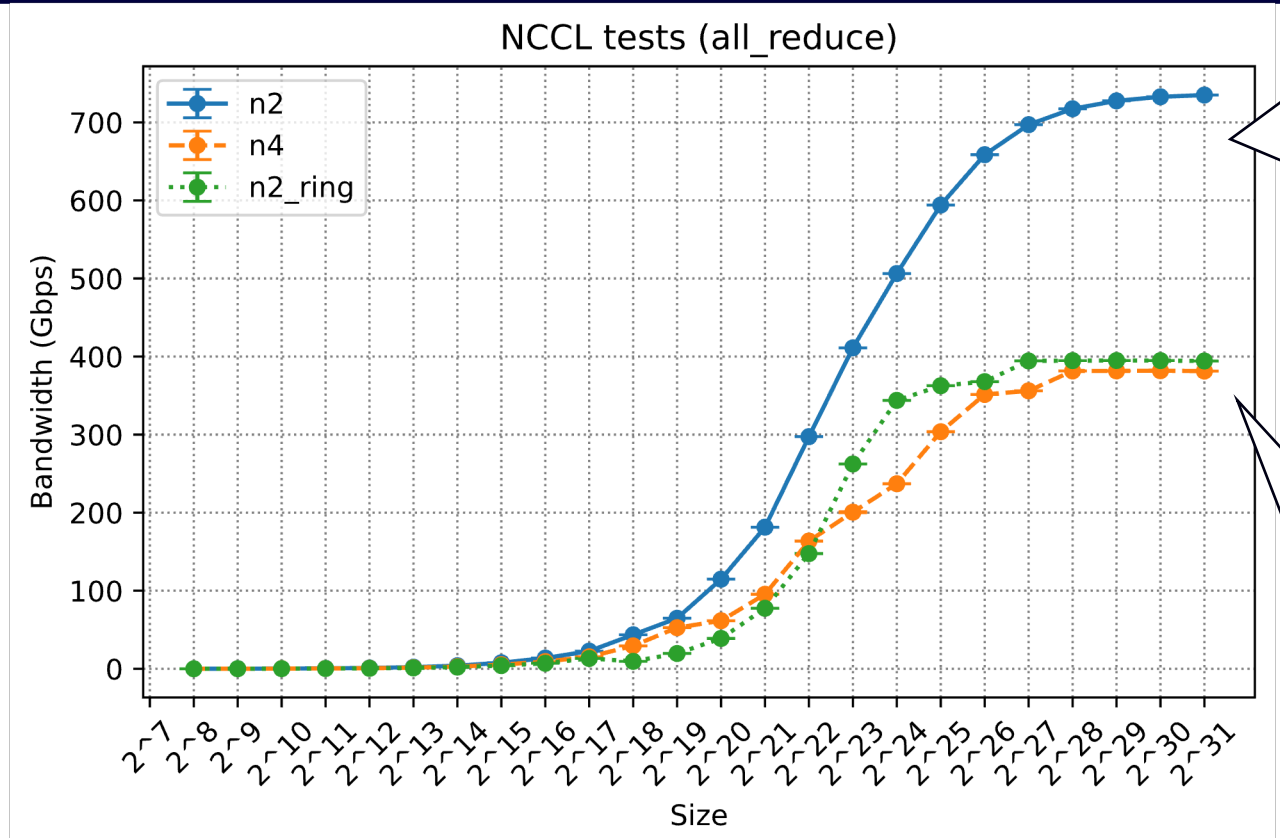
Node=4、NIC=2



Ring Algorithm利用時の性能

Ring Algorithmを強制した場合の帯域性能が想定より低い

AllReduce 性能測定結果（Ring Algorithm強制あり）

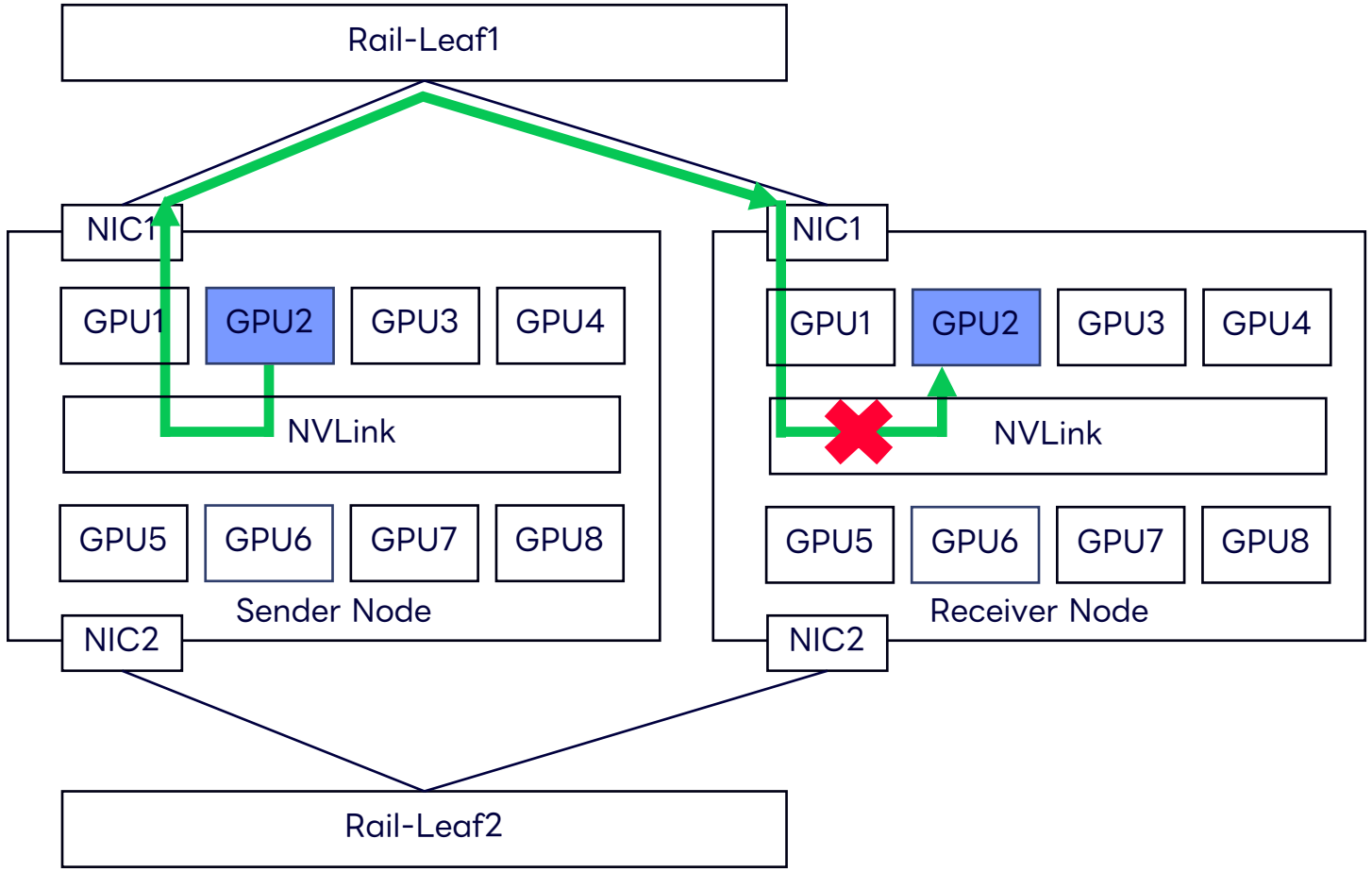


Ring Algorithmを使ってない場合
想定した性能が出ている
（NVLSTreeが選択されていた）

Ring Algorithmを使う場合
N=2の場合出会っても、
400Gbpsで律速された

PXNの当初の想定・誤解

PXNはSender Sideでしか機能しない



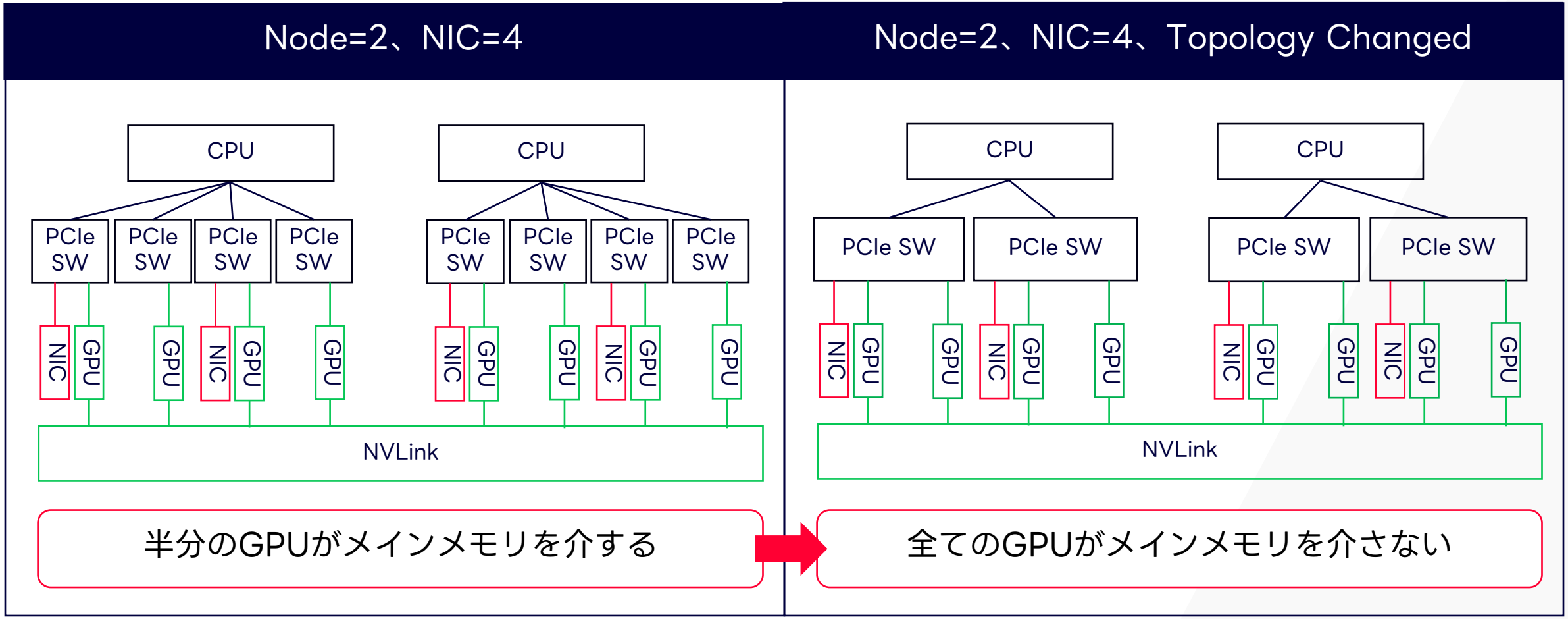
PXNはSender/Receiver両方で動作すると思っていた。

実際はPXNはSender側でしか動作しないことがわかった

NICの枚数を減らすと、Receiver側では受信したNICから対象のGPUにデータを送る場合は、GPU Direct RDMAができないGPUに対してはCPUを経由してしまう

NICの追加とFWの更新による影響調査

全てのGPUがNICと直接通信できることが性能を向上させるか？

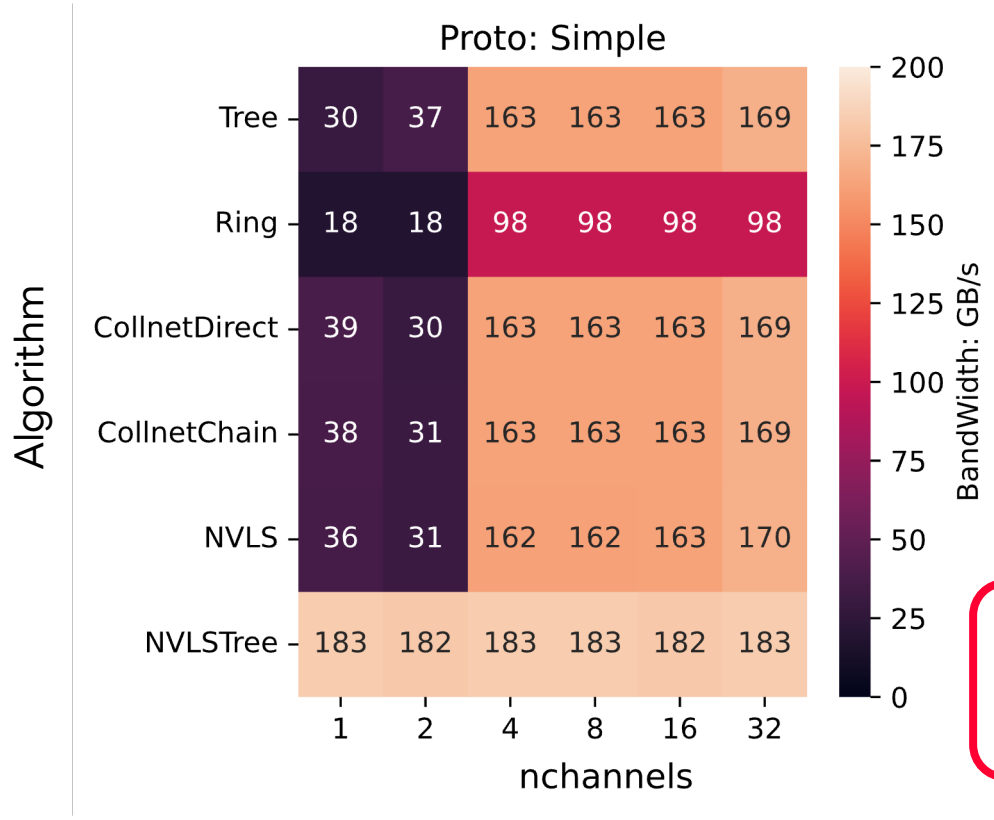


Ref:アクセラレータ間通信の実際 (https://mpls.jp/2024/presentations/mpls2024-PFN.pdf)

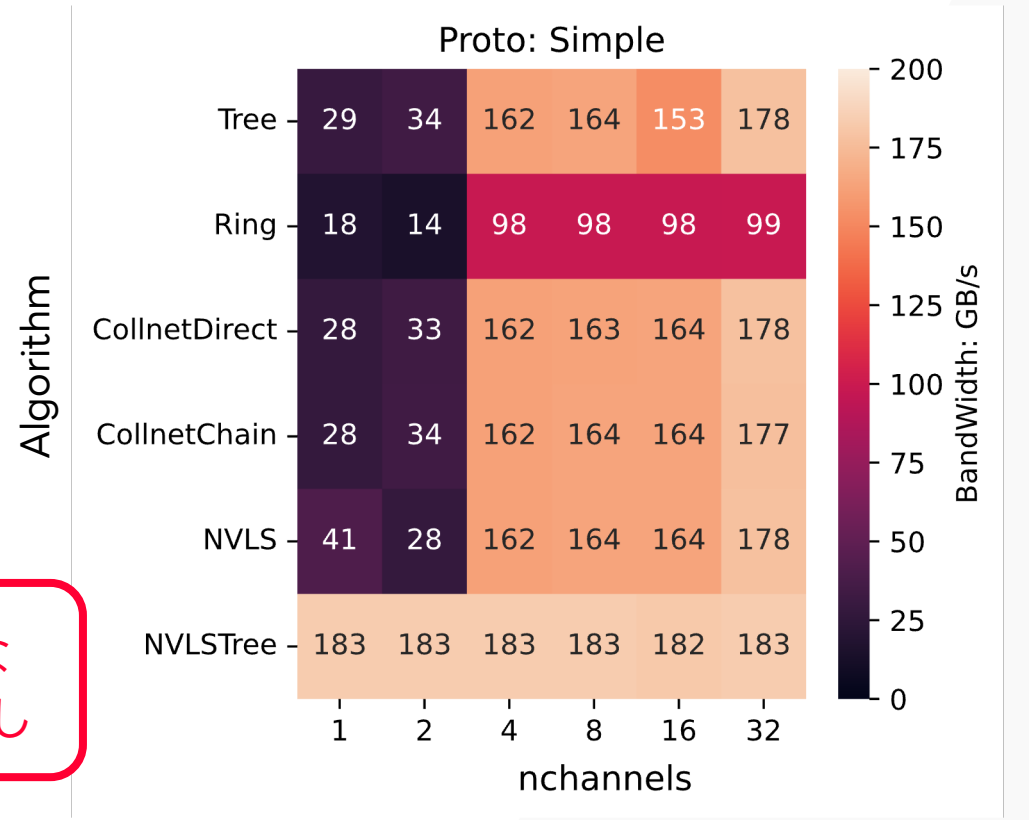
Server内Topologyの変化と性能変化

仮説とは裏腹に大きな性能変化はみられなかった

Node=2、NIC=4



Node=2、NIC=4、Topology Changed



大きな
変化なし

未知の領域に対してのエンジニアリング

予期せぬ自体が発生した際のマインド

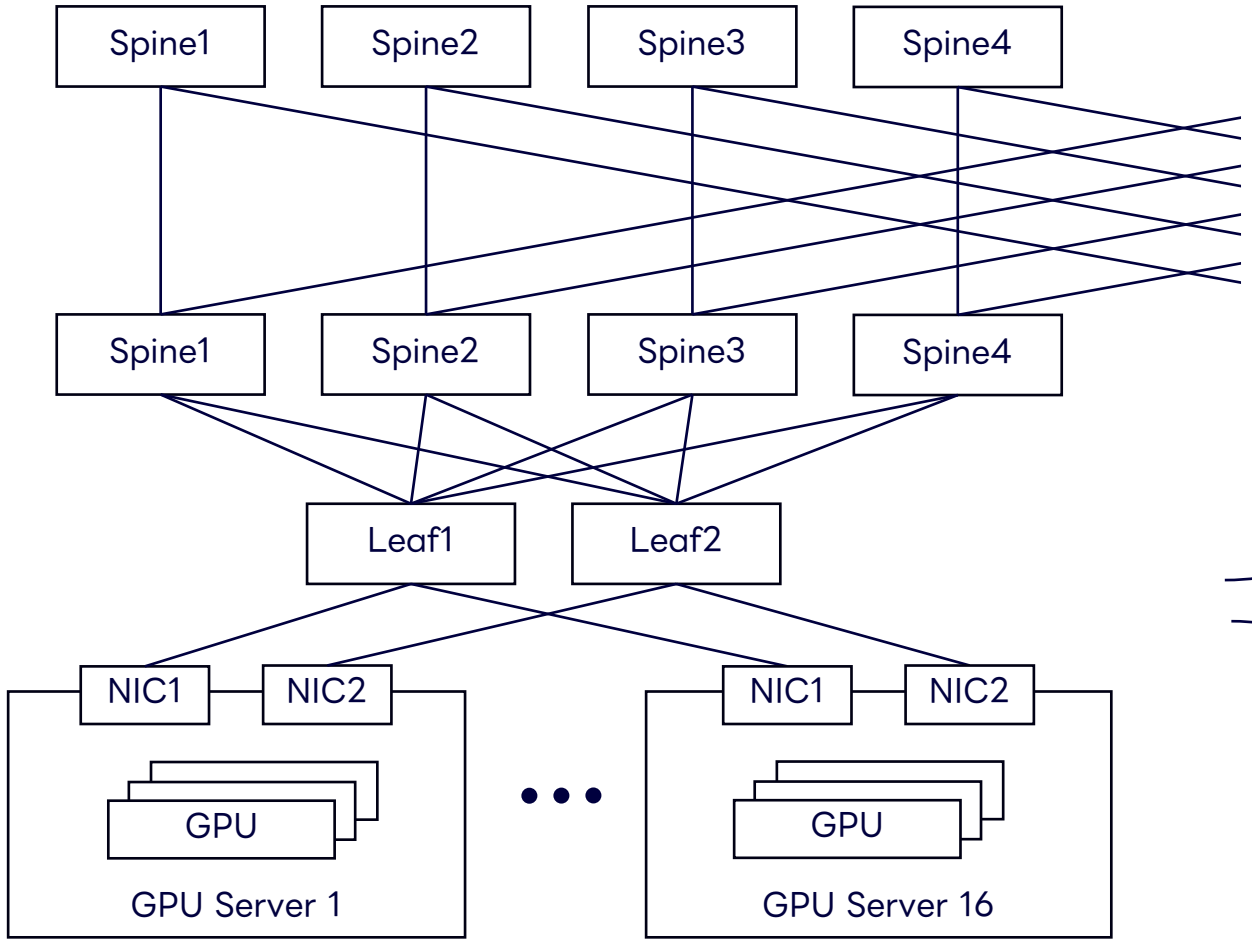
- 現時点では気になるものの致命的な問題ではない
    - 現状はまだこの性能で困るという状況では無いため、実際はほぼ無風
    - コスト削減とのトレードオフの前提はあるため、誤った判断だとも思っていない

- 詳細に性能測定などを行なっていった結果として見えてきた課題なのでポジティブに捉える
    - 様々な構成、Node数、パラメータ変更をして根気良く調査しないと見えてこない問題だった
    - 逆に学びの機会だと捉える。こういうつまづきがないと、必死になって調べない

新たに見えてきた課題と
今後の取り組み

全体最適化のための監視のあり方

E2Eの最適化にはE2Eの監視が必要になるのではないか？



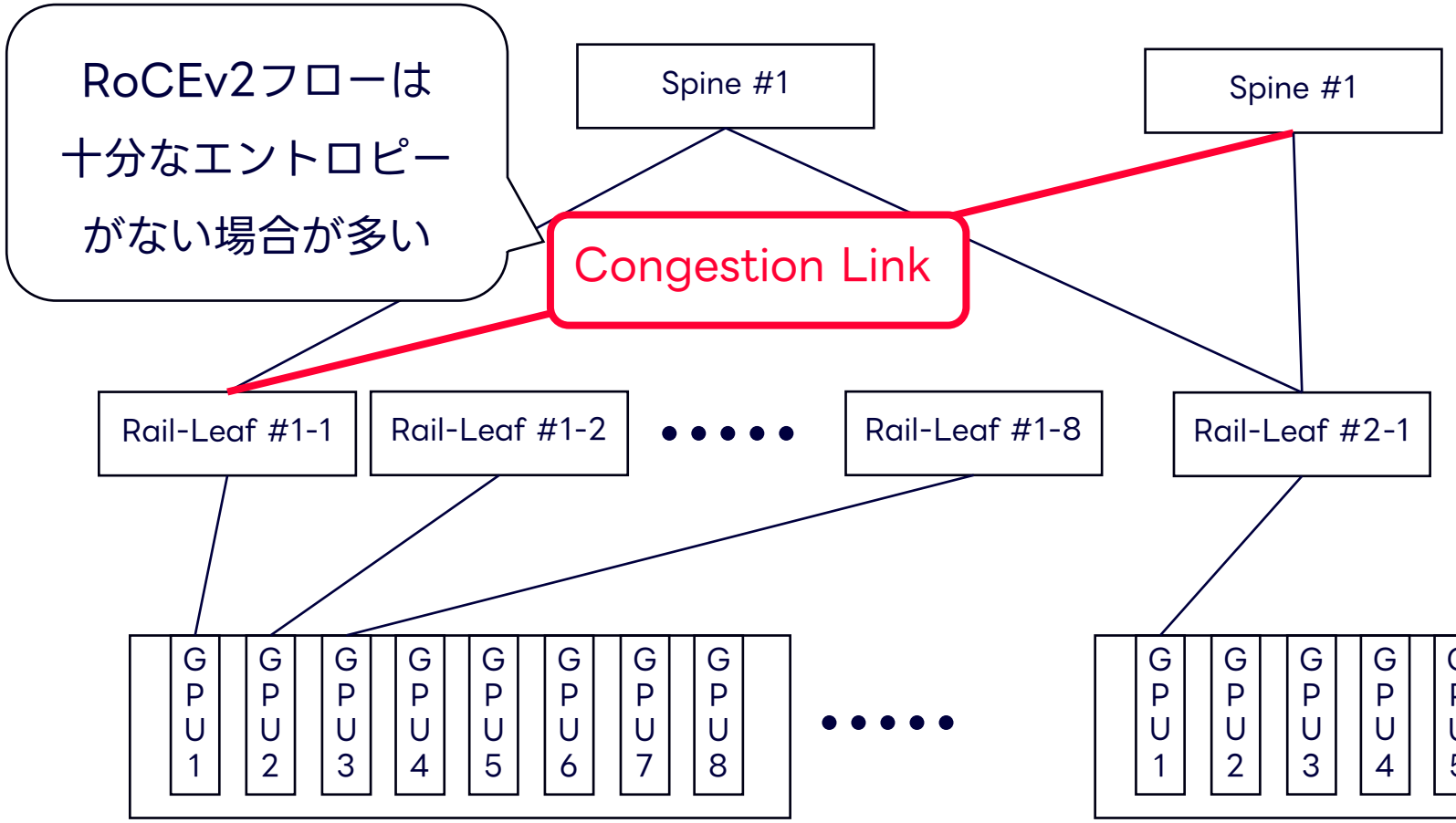
従来のNW監視
＋
GPU NWの追加監視
（PFC、ECN、…）

サーバの監視
＋
GPU NICの追加監視
＋
GPUの監視

全域の統合監視

負荷分散の問題

RoCEv2における5-tuple ECMPの問題とその他の負荷分散手法



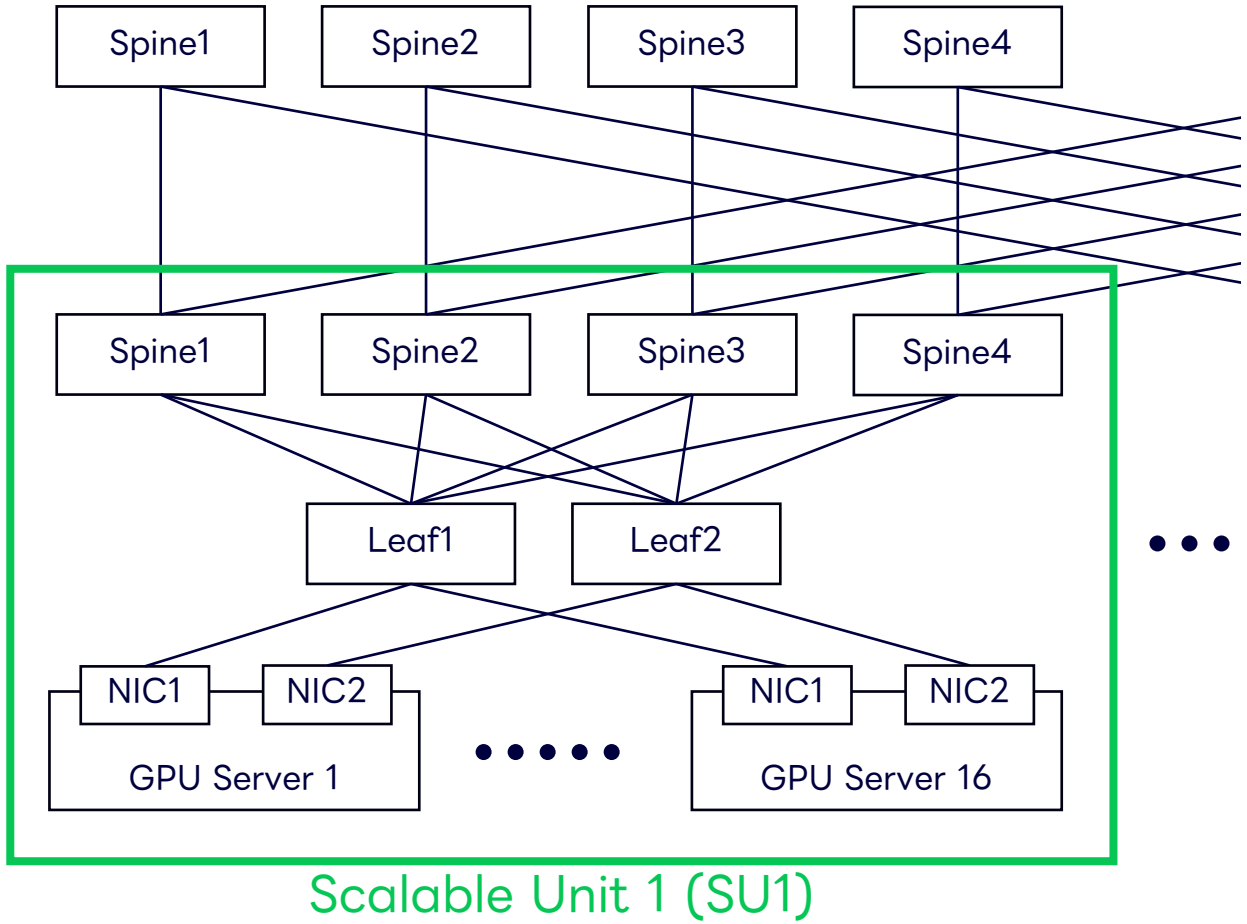
負荷分散方式と実装

- Flowlet Switching
  - DLB/GLB（flowlet mode）
- Packet Splaying
  - DLB/GLB (splay mode)
  - Adaptive Routing
  - Scheduled Fabric
- PXN前提で負荷分散しない

Ref: AI/MLデータセンターネットワークでの負荷分散手法　-NW運用者の視点から-
(https://www.janog.gr.jp/meeting/janog54/wp-content/uploads/2024/06/janog54-lt2-kobayashi-00-3.pdfhh)

LYでの負荷分散の今

スケジューリングの調整によるSU跨ぎ通信の回避



SUに閉じている場合はLeafで折り返されるため、負荷分散の必要はない

負荷分散を必要とするのは主にSUを跨ぐ通信

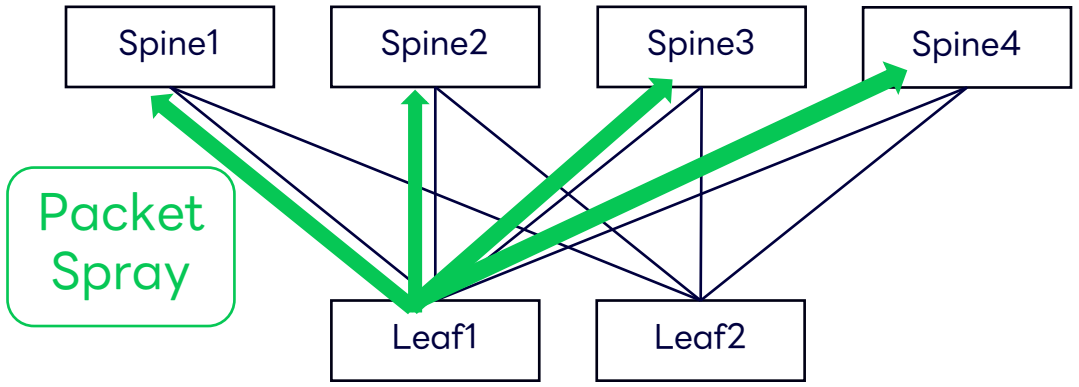
Podのスケジューリング先をSUに閉じるような形にし、負荷分散技術は利用していない

サポートできないのは16 x 8 = 128GPU以上使うケースだが、現状ここまで要求してくるユーザはいない

Ref:AI時代のデータセンターネットワーク(https://speakerdeck.com/lycorptech_jp/dcnw_in_the_ai_era)

Ultra Ethernet Consortium（UEC）

業界の動向と期待

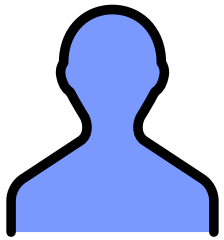


UECではSpray方式での標準化が進められている模様

RoCEv2と比べてスイッチに求められる実装が複雑化しそうだが、RoCEv2を完全に置き換えるのか、あるいはInfiniBandのように棲み分けになるのか？

GPUのスケジューリング改善

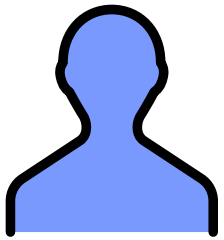
さまざまなワークロードを収容するPlatform特有の問題



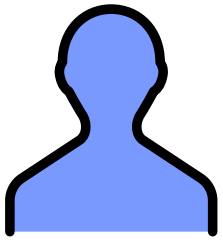
Tenant A

GPUをたくさん並べて
分散学習したい

分散学習にトライしたいけど
まずはお試しでやってみたい



Tenant B



Tenant C

現状は分散学習までいらない

そもそも社内には以下のような
ユーザ要求のグラデーションがある

- 高速NWが必要な人
- 小規模にトライしたい人
- 現状は単体Node上のGPUで十分な人

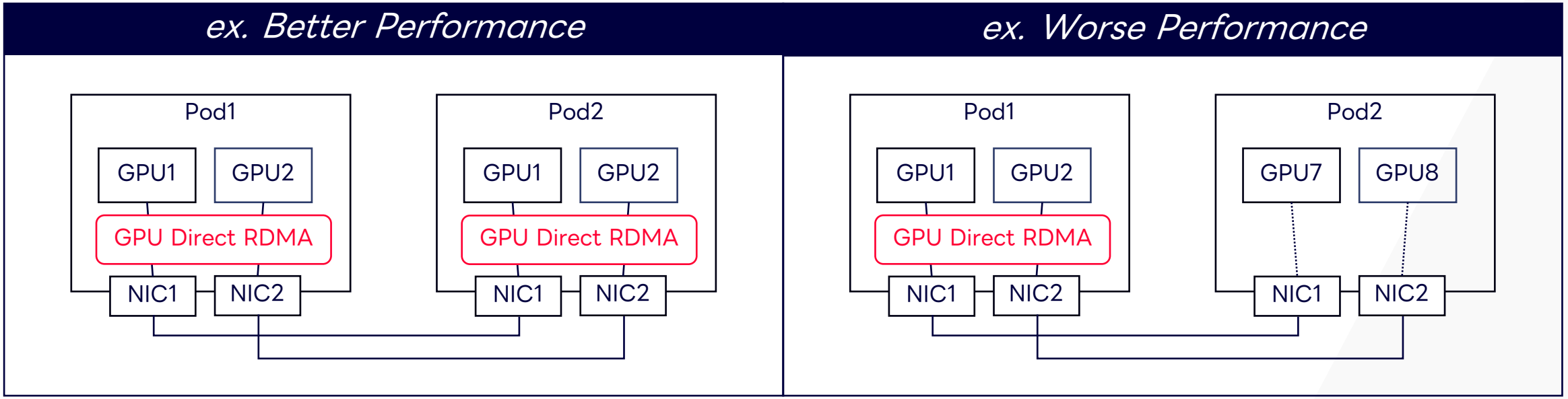
ACPでは、これらの要求に対して

- コストを最適化しつつ
- ユーザの要求には幅広くリーチし
- 同一Platformとして提供

ができることが望まれる

GPUのスケジューリング改善

NICの枚数を減らしたことによる性能担保の難しさとその対応



さまざまなパターンで傾向分析、NCCLパラメータ最適化を行ったところ、全てのパターンで良い性能になる設定はなかった

ユーザ提供のポリシーの検討

最適なスケジューリング方式の検討と実装

キャッチアップの難しさ

最先端技術の進化のスピードの速さとキャッチアップの困難性

- 国内での事例は少なく、事例ベースの情報収集は難しい

- 海外の先行事例を収集し、社内インフラに導入して嬉しいかを取捨選択する必要がある
    - 「生の声」の収集には、海外カンファレンスへの参加なども非常に重要
    - 各社がさまざまな技術について、ポジショントークに技術的に鋭い指摘を交えながら話すのでファクトチェックを行いながら整理して考える力が必要

その他、今後行っていきたいこと

800Gを見据えた知見の収集やDOCAエコシステムの検証など

- 800G Switchの検証
  - サーバは400G NICのまま、800G Switchの1ポートにQSFP-DD x 2を搭載する形
  - 既存の構成を保ちつつ、今後の進化を見越して先行して検証を進めていく

- DOCAエコシステムの検証
  - 特にメトリクスやテレメトリ機能などの確認や、場合によっては検証を行いたい。
  - High Frequency Samplingなどの機能があるため、マイクロバーストによる影響などが観測できるようになるのではないかという期待

まとめ・議論

まとめ

Rethinking AI Infrastructure: LINEヤフーが描く、内製技術で切り拓くネットワークとエンジニアリングの新時代

- Rethinking AI Infrastructure
  - GPUを中心としたコンピューティングシステムの全体最適化を行なっていく必要がある
  - ドメインを超えて、ボトルネックを特定し解決できるエンジニアが必要

- LINEヤフーの事例の紹介
  - 自分たちの課題解決や実現したいビジョンへの試行錯誤を外注しない
    - 実際にワークロードを持っている組織にしか、その問題を解決する答えはない
    - 問題の本質を捉え、最適な解決策を模索することが重要
  - **こういう取り組みこそエンジニアリング**

今後のAIインフラエンジニアに求められる姿

あなたはどう考えますか？

- エンジニアリングとは？
    - 技術を応用、適用して問題を解決すること
    - 問題の本質を捉えることと、適切な技術を選択することが重要
- 組織が直面している問題の分析には、単一領域にとらわれない幅広い視野が必要
- 適切な技術の選択には、その領域についての深い理解と探究力が必要

手を動かすしかないのでは？

議論

議論したいことの一例

- オンプレミスでAIインフラを持つことをどう思いますか？
- Ultra Ethernetはどうみていますか？何を期待していますか？
- データセンターネットワークは今後どのように変化していくのか？
- この先ネットワークエンジニアに求められるものは何か？
- …etc