NFVの商用化の開発と運用の経験から将来のネットワークアーキテクチャを考える

株式会社NTTドコモ
ネットワーク開発部
中島 佳宏

2020/10/15

ネットワーク仮想化によるコアネットワークの進化

無線方式の進化と共にコアネットワークも進化
ネットワーク仮想化（NFV）により汎用ハードウェアを効率的に利用

交換機

aTCA

IAサーバ

アナログ交換機

デジタル交換機

ATM交換機

All IP ネットワーク

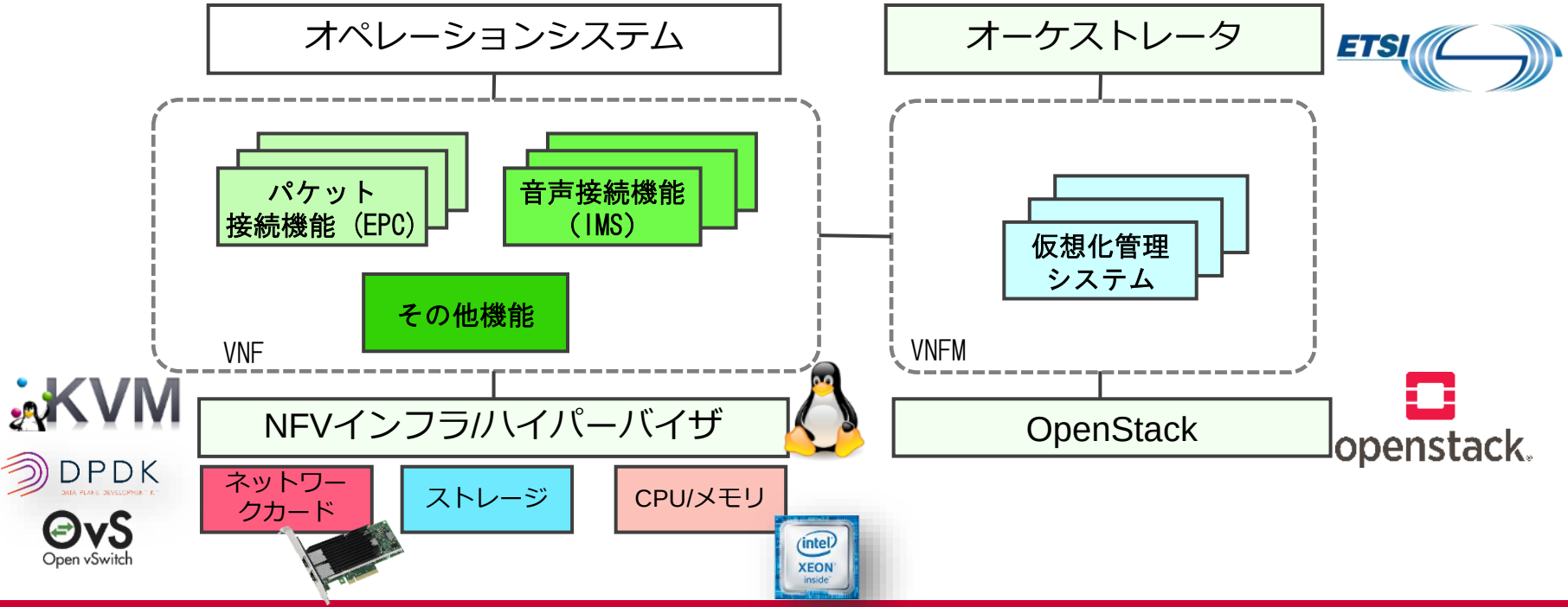
仮想化ネットワーク

1980年代 | 1990年代 | 2000年代 | 2010年代 | 2020年代

1G | 2G | 3G | 4G | 5G

ドコモのネットワーク仮想基盤の構成

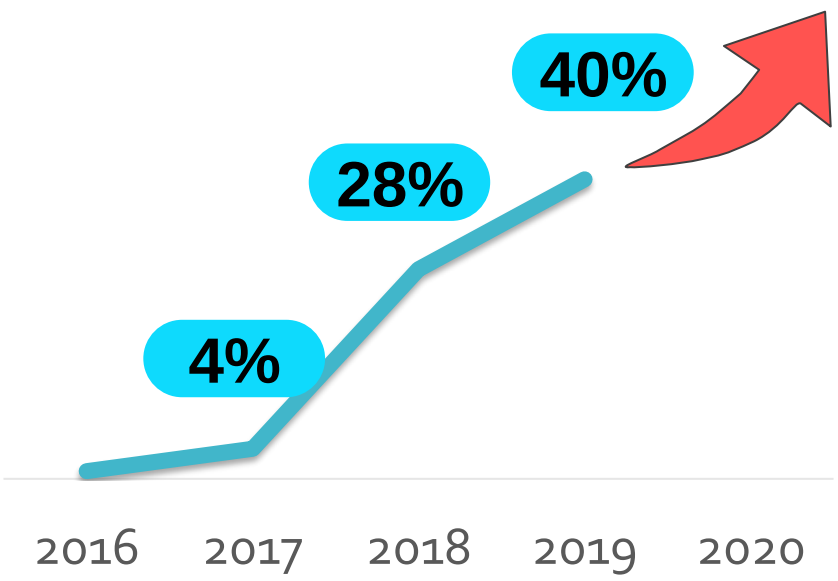
標準化準拠のアーキテクチャでオープンソース、汎用ハードウェアを
活用したマルチベンダ対応のネットワーク仮想化基盤を構築



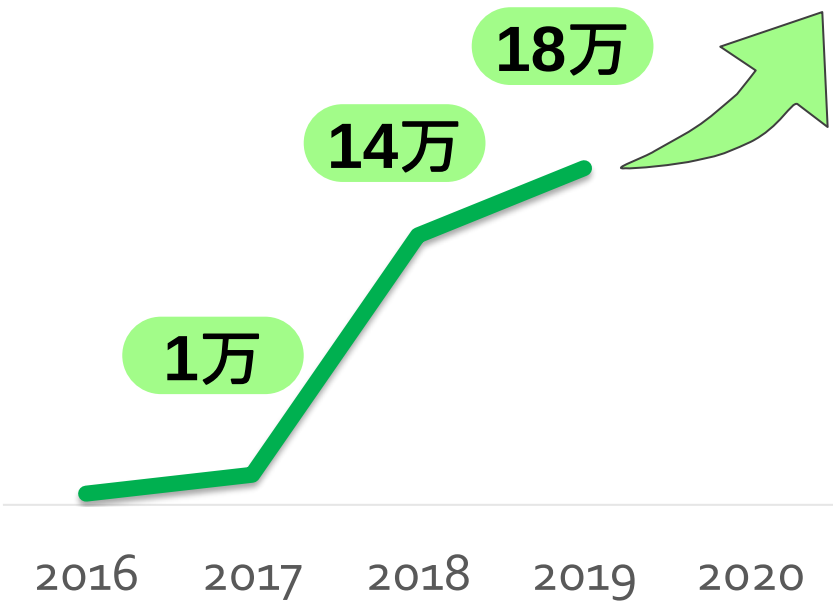
ネットワーク仮想化の導入状況

仮想化システムの増加と全国展開にむけ適応率と規模も拡大中

仮想化比率



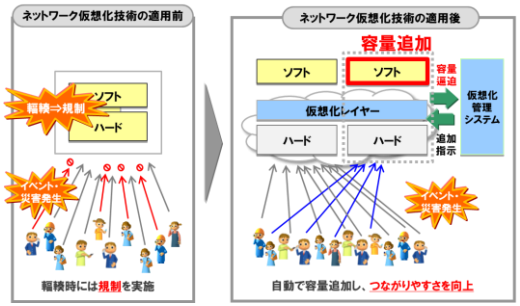
仮想化基盤の規模（ｖCPU）



ネットワーク仮想化導入により得られたメリット

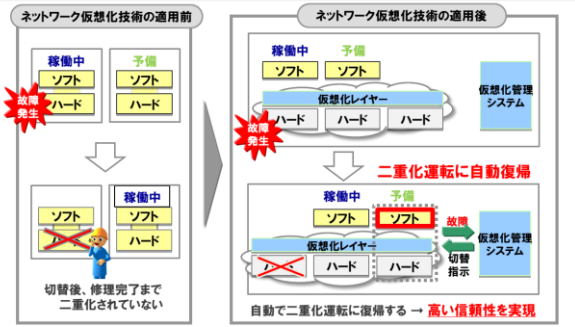
メリット① 通信混雑時のつながりやすさの向上

・自動的に通信設備の容量を追加し、つながりやすさを向上



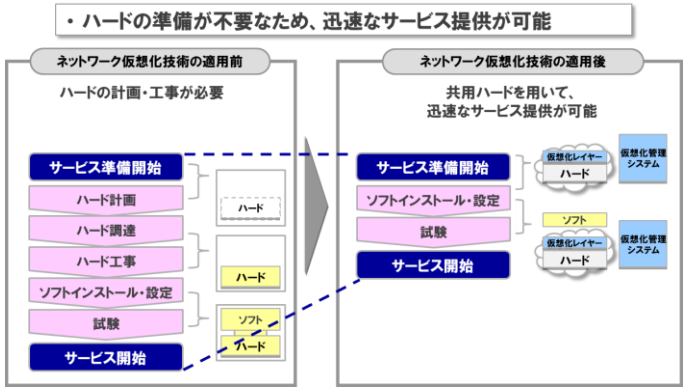
メリット② 通信サービスの信頼性向上

・ハード故障時に、二重化運転への自動復帰により高い信頼性を実現



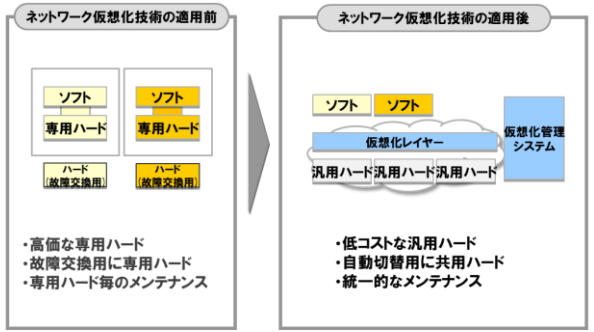
メリット③ サービスの早期提供

・ハードの準備が不要なため、迅速なサービス提供が可能



メリット④ ネットワーク設備の経済性向上

・汎用ハードが共用可能となり、ネットワーク設備の経済性を向上



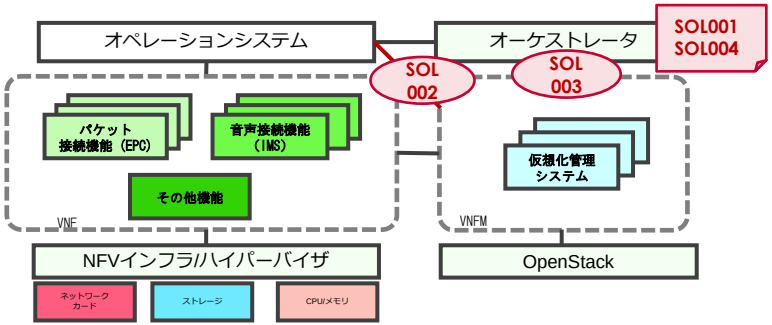
ネットワーク仮想化基盤の現在

通信事業者として“通信サービスを継続したまま"
仮想化基盤やNFVO/VNFMのアップデートを完了!

OPSシステムと連携しVNFのサービスを継続したままのVIM/NFVIのアップデート

旧バージョン　アップグレードのためのヒーリング　新しいバージョン上にヒーリングでVMを戻す

ETSI NFVのStage-3仕様に準拠したIFを用いてNFVO/VNFMのアップデート

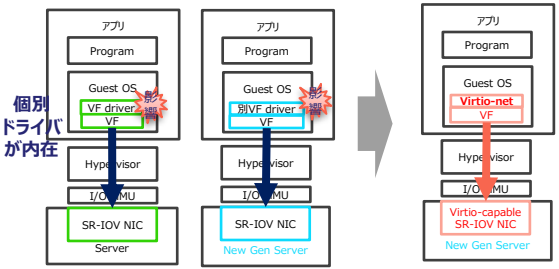


- SOL001 (VNF Descriptor)
- SOL002 (Ve-Vnfm)
- SOL003 (Or-Vnfm)
- SOL004 (VNF package)

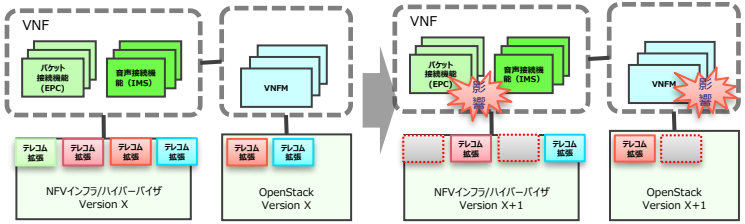
ネットワーク仮想化の商用開発と運用の経験

オペレータとして長期運用やアップグレード・マイグレションを見据えた アプリと基盤含めた全体設計や先行的な検討はとても重要

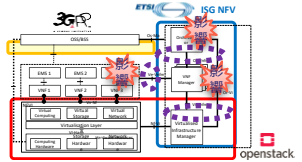
ソフトウエアとHWとの分離は さらに促進すべき



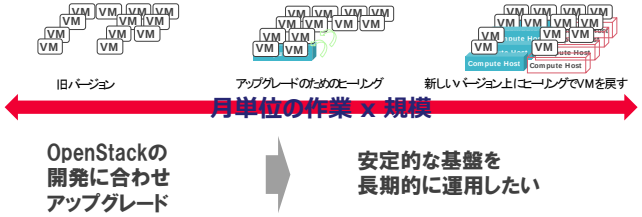
黎明期のテレコム拡張は将来のアップグレードのリスクに変化，メジャー機能化の推進または特殊機能は制限すべき



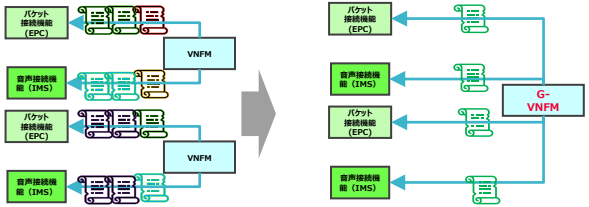
カスタマイズから標準やデファクトへの移行は大変



アップグレード作業の迅速化・効率化の促進や，仮想化基盤を長く使えるようにする設計や取り組みは重要



アプリ個別のきめ細やかな制御より，個別アプリのVNFに共通で使えるシンプルな管理制御への移行が重要



今後のNW仮想化の高度化にむけて

ネットワーク仮想化の効果最大化に向けた課題にとりくむ

経済的なNW
開発・検証・設備の
費用の効率化

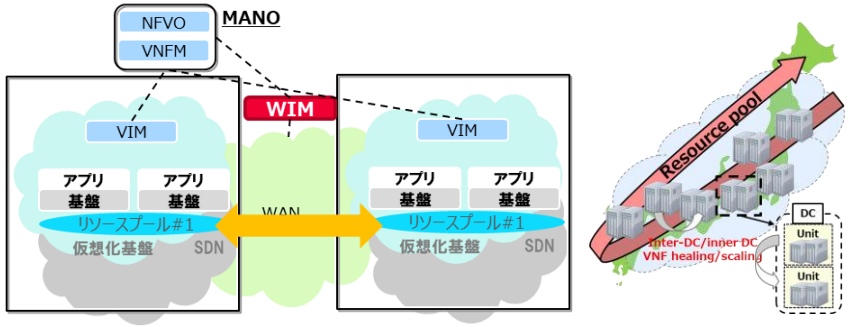
災害・故障に強いNW
全国リソースを活用した
サービスの高可用化

APLの進化を支える基盤
zero-touch化
クラウドネイティブ対応

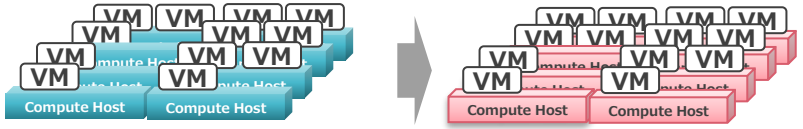
サービスの迅速な提供
開発の短期間化

依存性の排除	仮想リソースのダイナミックな移動	進化支援のためのAPI高度化	開発効率化	高性能化

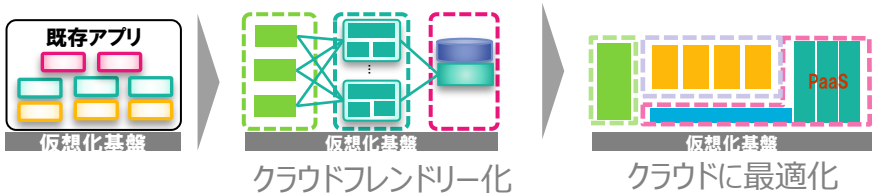
VIM/局舎連携による全国リソース共有と柔軟な仮想資源の移動



基盤アップグレードの短縮化



アプリケーションの進化



クラウドフレンドリー化　　クラウドに最適化

5G時代のコアネットワークで実現したいこと

経済性の追求

- スケールメリットを生かしたハードウエアやソフトウエアの経済化
- インフラ設備として省電力・省スペース化、高収容・高性能なノードの実現
- ベンダ都合のEoLやEoSの回避と設備ライフサイクルの長延化
- 新規機能開発のための開発範囲の局所化・開発期間の短縮・検証の自動化

信頼性の向上

- 故障からの自動復旧と故障時計画保守の実現
- 急激なトラヒック需要変化に対する柔軟な容量の追加・削減
- 高い可用性と冗長化

自動化・保守運用の効率化

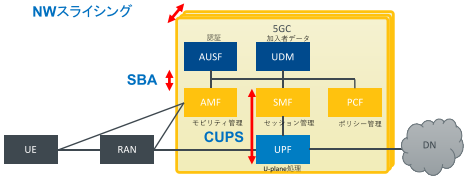
- 構築や構成変更の容易化
- アプリ種別によらず統一的な保守運用方法の実現 (サービス個別の保守方法の削減)
- 自動化促進によるマニュアル作業の削減・現地作業の削減
- 改修やアップグレード作業の短期間化・容易化

サービスの早期提供

次世代NWアーキテクチャの検討のねらい

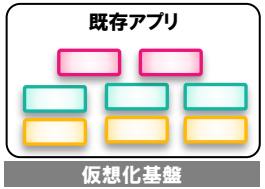
ドコモの仮想化基盤の課題を解決しネットワーク仮想化の効果を最大化する次世代NWアーキテクチャの検討をすすめている

5GCでのアプリの要件分析



5G時代のアプリの要件・最新のクラウド技術・コンテナ技術を踏まえた検討

仮想化における課題と解決の方向性



ドコモでの仮想化基盤の商用経験やノウハウを活かし，
仮想化の課題の解決や効果最大化に向けた技術検討

次世代NWアーキテクチャの策定

クラウドフレンドリー化

クラウドに最適化

ドコモ（オペレータ）の課題を解決するための
仮想化基盤とアプリの両方のアーキテクチャを検討する

次世代ネットワークアーキテクチャの実現にむけての検討項目

従来からのテレコム要件のベースラインの実現を確保しつつ
5G時代に求められる新機能の具備・
構築や保守の高度化・効率化の実現をめざす

NW設計:
既存ノードとインターワークしたうえで
仮想化・コンテナ化にむけて
なにをかえないといけないのか

規制制御:
アプリの進化に伴い
変える必要があるのか

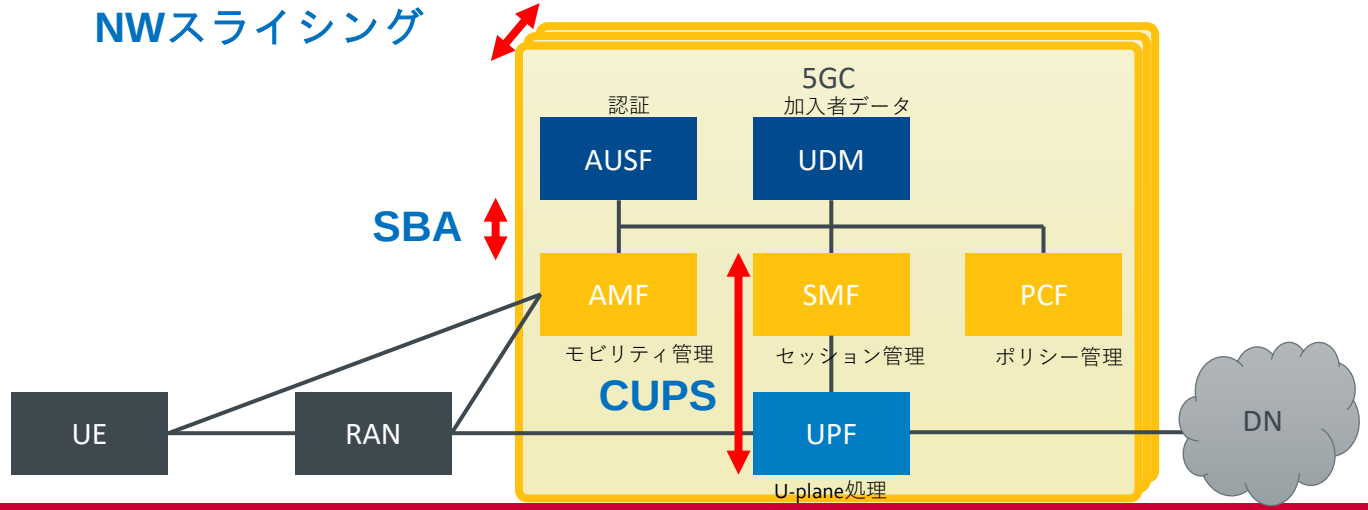
保守:
アプリのコンテナ化・マイクロサービス化への
対応や基盤の保守にむけて何を変えるのか?

アプリのアーキテクチャ:
従来のテレコム要件を満たしつつ
5GCの新しいアプリ要件を
どう実装すべきか?

基盤:
5GCの新しいアプリ要件
(特にコンテナ・コンテナオーケストレーション)
にむけて何を対応しないといけないか

EPCと親和性の高い技術をもちいて柔軟にNWを構築

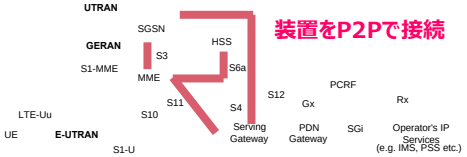
1. SBA (Service Based Architecture): 機能をサービスとして定義
2. CUPS (C/U-plane separation):
   制御信号処理とパケット処理を分離·柔軟な配置
3. NWスライシング: 複数QoSの異なる個別NWを提供



4G時代のコアNWアーキテクチャから5GCアーキテクチャへの変更点

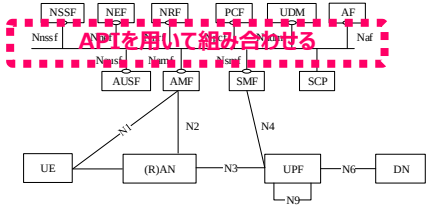
装置をPoint-to-Pointで接続しサービスを実現から，ソフトウエア機能部をAPIで接続しサービスを実現へ

テレコム専用プロトコルの活用



汎用的なユースケースに基づくノード配置

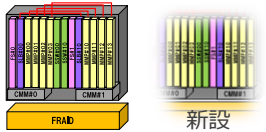
Web系プロトコルでテレコム要件を実現



NW機能のを自由に組み合わせ･カスタマイズ

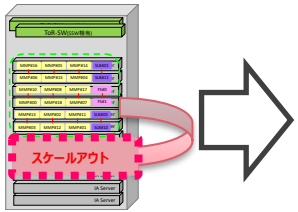
スケールアップ指向から，ブレード (VDU) ごとのスケールアウト指向をへて，より細かい機能レベルでのスケールアウト指向へ

収容規模が装置自体に束縛



EPC/aTCA

ブレード (VDU)レベルのスケールアウト



EPC/NFV

API提供レベルでのスケールアウト

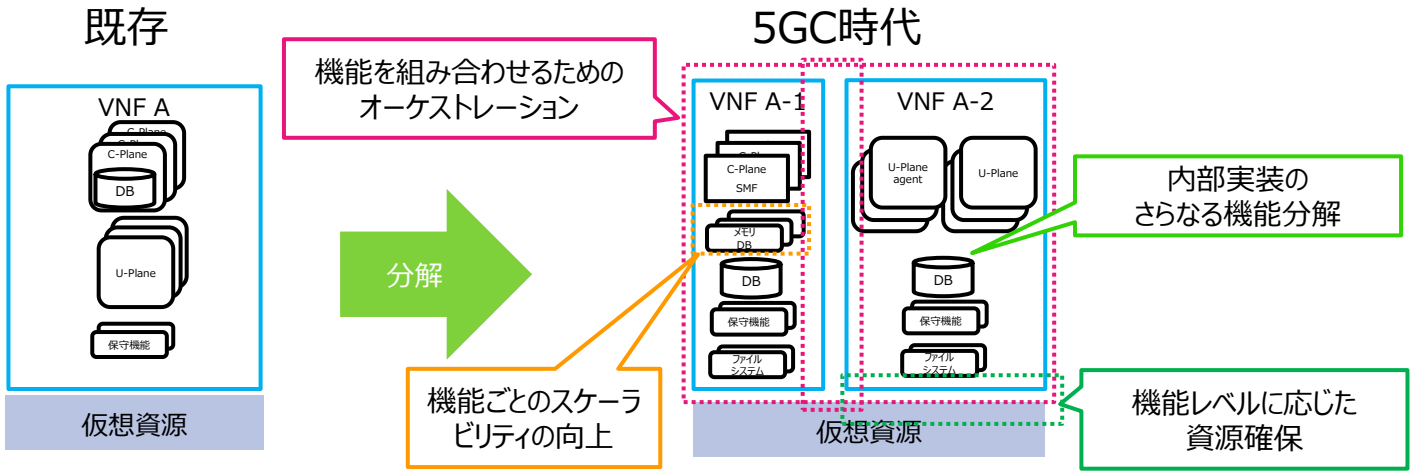
5GC/クラウド

仮想化技術の活用を前提に
アプリの実行環境を動的に増減設
(オーケストレーション)

柔軟に収容数を変更可能なスケールアウト型のNWアーキテクチャへ

5GCアーキテクチャ変更に伴う新たな要件について

1. NFでのより細かい機能 (API) レベルでの管理・保守機能
2. 各NFを組み合わせサービスとして実現するための連携管理とその資源確保
   (サービスや資源のオーケストレーション)
3. Web系開発スタイルやデプロイ方式への対応
   (コンテナ環境やコンテナオーケストレーションの対応)



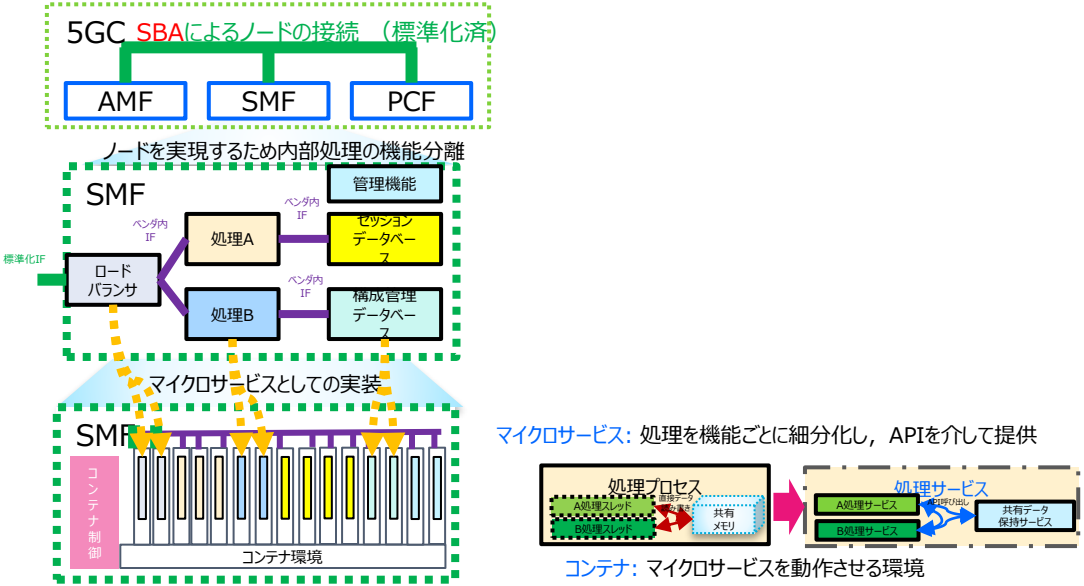
5GCのキーワードと実装の関連性

5GCのノードレベル

ノードの内部の
機能分担レベル
（アーキテクチャ）

ノードの実装レベル
(コンテナ環境)

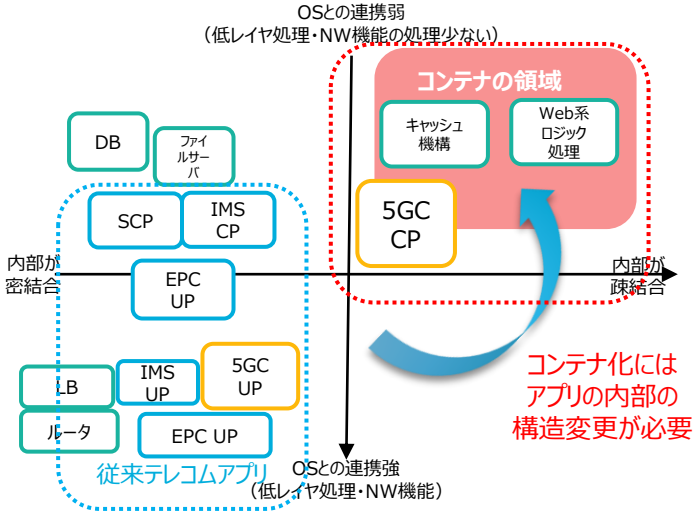
ハードウエア含めた
ノードの実装レベル



テレコムアプリのコンテナ化の考察

テレコムアプリの処理については，本来コンテナが不得意な領域であり，
アプリのコンテナ化への改造やコンテナ基盤自体の拡張を行う必要あり

コンテナの適応範囲



テレコムアプリ収容にむけたコンテナ基盤の拡張

必須の拡張機能
1. 性能安定化のための拡張
2. NW処理アクセラレータ対応拡張
3. 複数NWサブネット収容拡張
4. テレコムプロトコル対応拡張

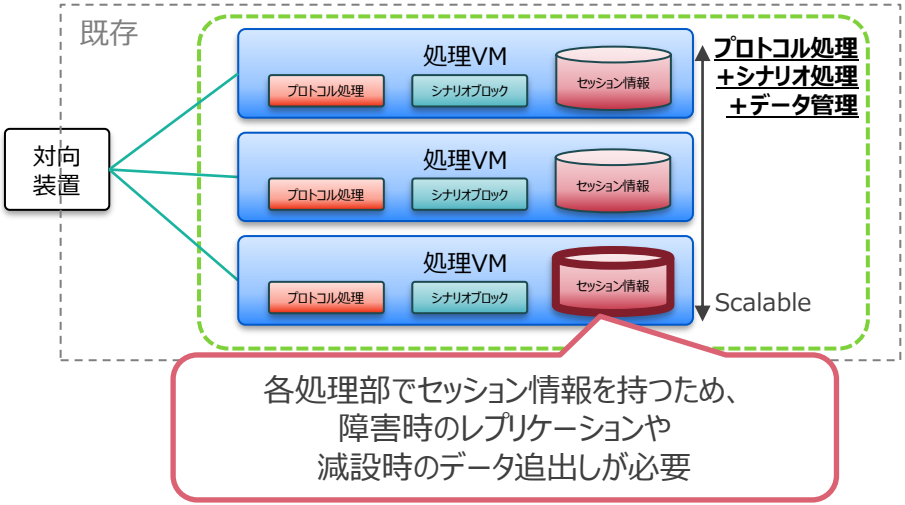
テレコムアプリのベースラインの要件とコンテナアプリ開発の考え方と対応

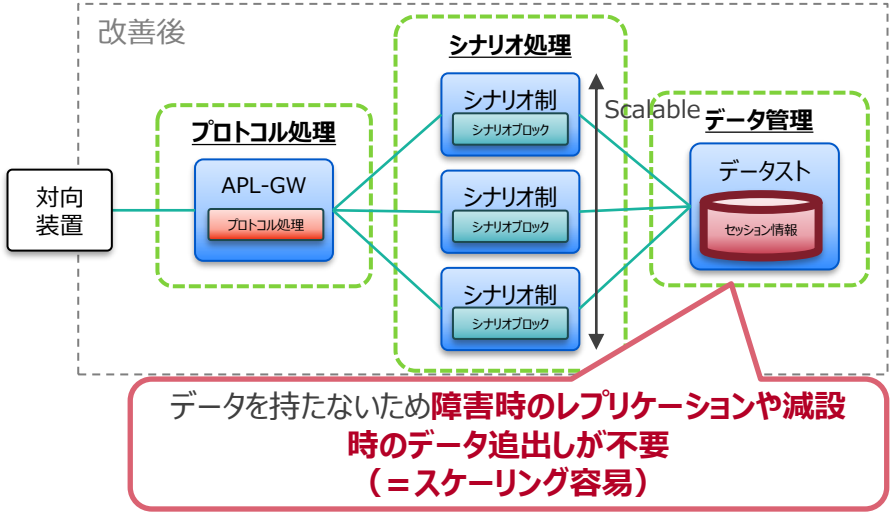
コンテナ環境をテレコムに合わせ手を入れるか or アプリを改造すべきか or VMとのハイブリッド構成にすべきか?

主なギャップ項目			一般的なコンテナの状況		右の要件を実現するための手段		
大項目	中項目	テレコム的な詳細	対応可否	デフォルトのコンテナの制約	コンテナ拡張	VMで対応	アプリの拡張
機能	処理内容	ステートフルな処理が必要	プラグイン対応	ステートレスを中心に対象 ステート情報はPaaSで	○	○	-
		複雑なアプリ構成	アプリ対応	シンプルなアプリを対象	-	○	○
	NW	複数IF対応	プラグイン対応	1POD 1IF	○	○	○
		処理オフロード機能対応 (SR-IOV, DPDK)	プラグイン対応	オフロード機能は非対応	○	○	-
	対象プロトコル	TCP, UDP, ICMP, SCTP, VxLAN,	α機能	TCP, UDP, ICMPのみ	○	○	-
	死活監視	インラインNWでの死活監視	アプリ対応	HTTPのGET要求で死活監視	-	-	○
非機能	性能	高いスループット	プラグイン対応	ヘビーなトラヒック量は想定外	○	○	-
		即時切り替え	プラグイン対応	10秒から分レベル	○	○	-

アプリケーションの内部の改善の考察 (1/2)

信頼性や性能などのテレコム要件を考慮し，
機能分離を図りモノリスから分散処理指向のアーキテクチャへ改造

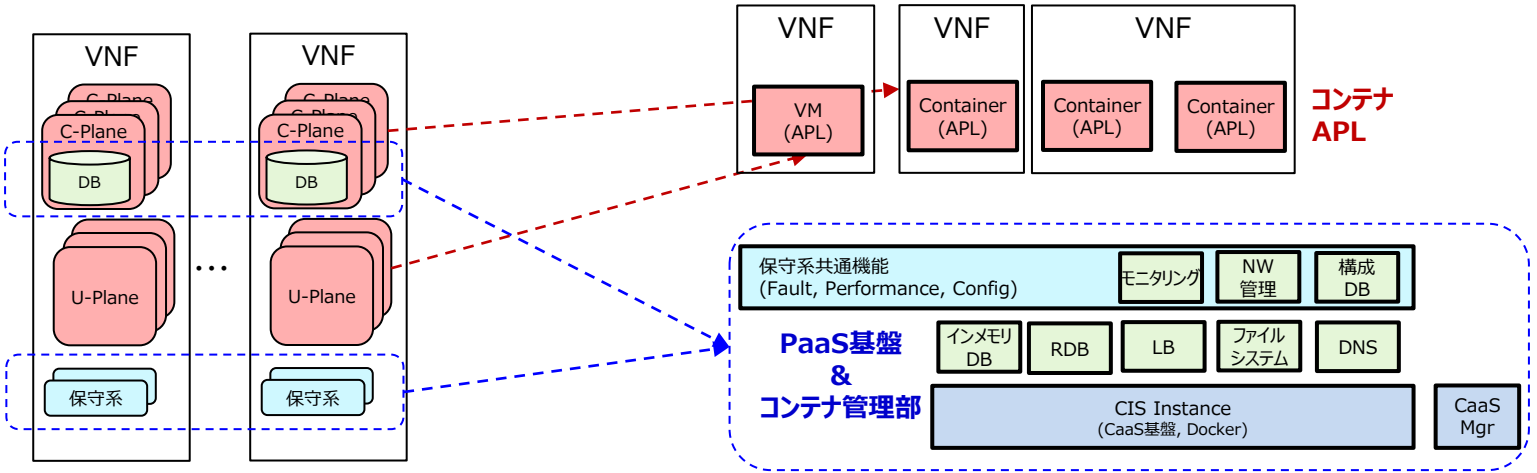




アプリケーションの内部の改善の考察 (2/2)

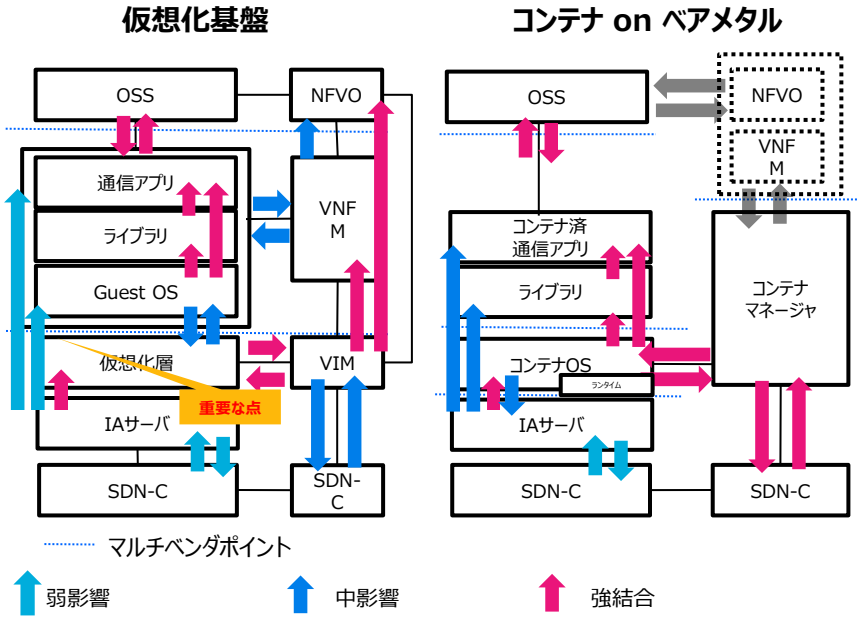
WEB系で用いられるスケーラビリティの向上のベストプラクティスを参照し
アプリ観点での変化へのアジリティ向上とと機能分離を図る

各VNFが持っていた共通機能はPaaSとして複数のVNFへ提供を図る



基盤のライフサイクルも考えないと行けない

開発のアジリティ向上のためのコンテナアプリ側に注目しがちだが，
更新頻度が高くなる基盤ソフトウエアの保守・運用と
長期的なサーバ導入や環境構築に向けた考慮は”すごく重要”



ソフトウエアのEoSupport

種別	更新期間	EOSまでの時期
仮想化基盤	0.5年	数年
コンテナオーケストレーション	0.5	半年-1年

HWのEoSales

種別	更新期間	EOSまでの時期
サーバ	1.5年	3-5年
NW DC SW	3-5年	3-5年

アーキテクチャの策定に向けた実装の方向性 (1/2)

コンテナのいいところと仮想マシンのいいところを組み合わせる

コンテナ環境のプラグインはメジャーなものにとどめ,
コンテナで実装するためのコンテナ環境のテレコム拡張は可能限り避ける

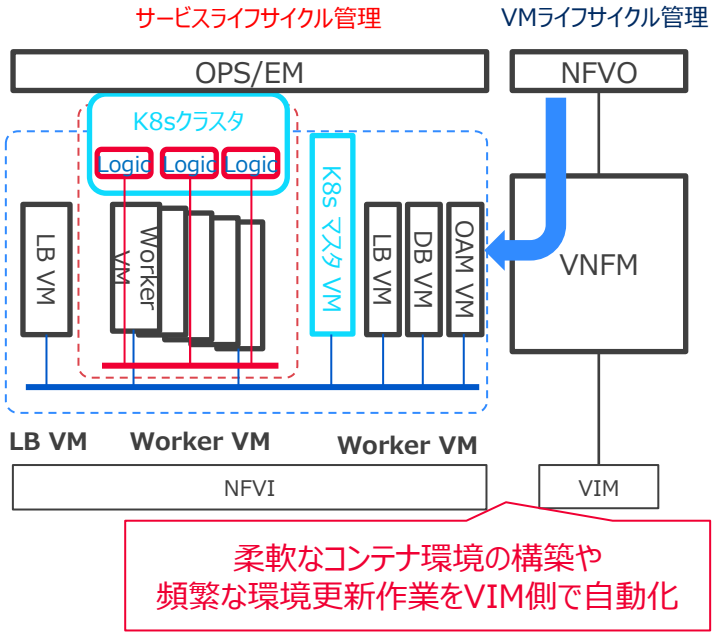
主なギャップ項目			解決の方向性
大項目	中項目	詳細	
機能	処理内容	ステートフルな処理が必要	ステートフルな処理はVMで実施
		複雑なアプリ構成	アプリをコンテナの流儀で実施
	NW	複数IF対応	複数IF対応はVM
		オフロード機能対応(SR-IOV, DPDK)	オフロード処理は基本VM
	対象プロトコル	TCP, UDP, ICMP, SCTP, VxLAN,	コンテナが対応しない処理はVM
	死活監視	インラインでのNWに対応した死活監視	外部監視装置を導入
非機能	性能	高いスループット	高い要求はVM
		即時切り替え	高い要求はVM

アーキテクチャの策定に向けた実装の方向性 (2/2)

5GCの要件と現在のコンテナ技術の成熟度を考慮し，
仮想マシンと仮想マシン上のコンテナ基盤を提供するハイブリッド型のアーキ

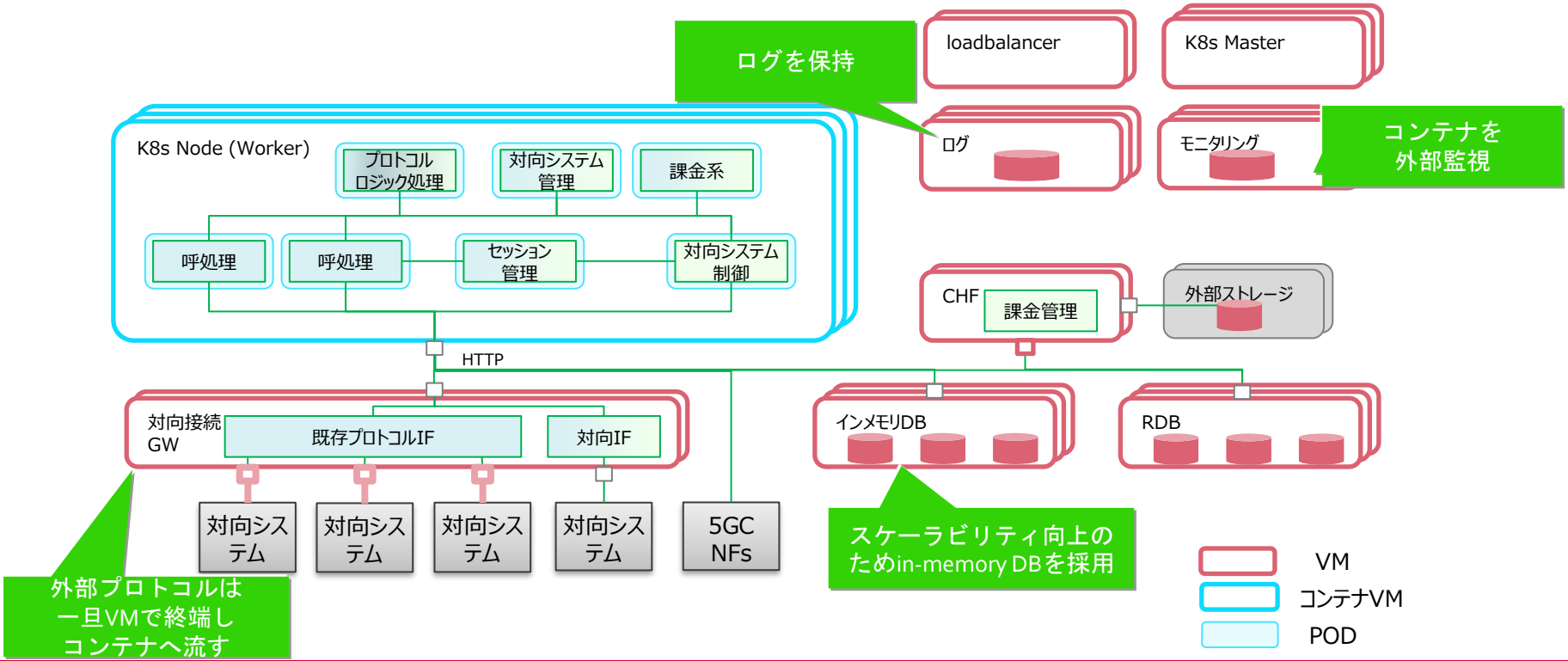
コンテナ基盤のアジリティも向上させ
仮想化基盤上で従来のアプリに加え5G時代のアプリも収容可能

処理種別	実装方式
SBAの実装やスケール性を求める処理 (コンテナに適した処理)	コンテナ on VM
外部との接続のために複数IFやプロトコル収容の処理 (loadbalancerの機能)	VM
瞬時の切り替えなど信頼性が要求される処理 (loadbalancerの機能)	VM
データやファイルの永続性を保持することが必要な処理 (PaaSの機能)	VM



提案する次世代NWアーキテクチャの全体像

テレコム要件を実現するため
コンテナのいいところと仮想マシンのいいところを組み合わせる



プロトタイプ実装による動作検証 ½ 性能観点

次世代NWアーキテクチャを具現化したプロトタイプでPoCを実施，
既存と同等の機能やSLAを達成していることを確認

既存アーキテクチャと比較しより効率的な処理が達成できていることを確認

観点	項目	提案アーキでの達成
設備観点	増減説のための 最小リソースの単位	◎ VMよりも 細かい粒度で増設が可能
	コアあたりの性能	○ 改善
保守運用 効率	インスタ	○ 従来同等の時間 (NF単位)
	スケールアウト	◎ かなりの改善 (Podの追加)
	スケールイン	○ 従来より改善
	完全復旧時間	◎ かなりの改善 (NF単位)

観点	項目	提案アーキでの達成
接続品質	性能ボトルネック	○ 特になし
	レイテンシの悪化	○ 特になし
HW障害時 の影響	不完了呼発生 確率	○ 従来同等
	不完了呼発生 時間	○ 従来より短縮

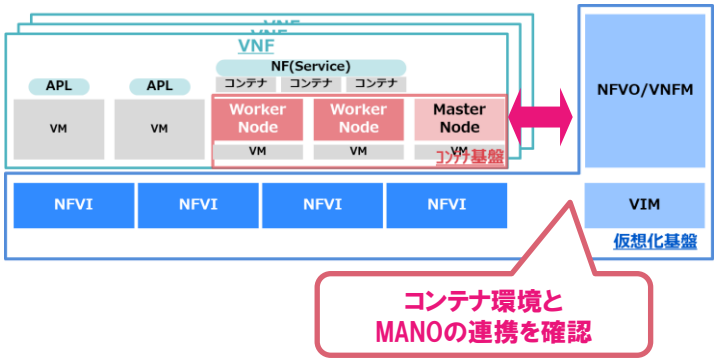
プロトタイプ実装による動作検証 2/2 保守・ライフサイクル観点

コンテナ on VMの形態も含む基本ライフサイクルシーケンスを策定し，
そのフィジビリティを仮想化基盤上で確認

基本的な障害ケースにおいて，VM上のコンテナ基盤(CaaS/k8s)と
仮想化基盤(MANO)の連携による復旧動作が可能であることを確認

【検証項目と結果】

検証項目	確認結果
VNFのライフサイクル基本動作	インスタ・ターミ・スケーリングに関して、問題なく動作することを実機確認
障害	障害を発生させ，問題なくヒーリングが動作可能であることを確認
輻輳	基本的な輻輳を発生させ，スケールアウトが動作すること確認



本アーキテクチャで改善される観点のまとめ

経済性の追求

- 従来よりも効率的な処理を達成

信頼性の向上

- 迅速に故障からの自動復旧と故障時計画保守の実現
- 急激なトラヒック需要変化に対する柔軟な容量の追加・削減時間の短縮化
- 完全復旧などの時間短縮や高い可用性と冗長化

自動化・保守運用の効率化

- 構築や構成変更の変更が容易化
- アプリ種別によらず統一的な保守運用方法の実現
- 自動化促進

サービスの早期提供

- 新しいコンテナ基盤を導入せずにコンテナ化されたアプリの導入も可能

まとめ

- ✓ これからもネットワーク仮想化の効果最大化や5G時代にむけ，
  5G時代のネットワークアーキテクチャやネットワーク仮想化技術の進化へ
  取り組みます

- ✓ 新しいネットワークアーキテクチャの検討をもとに
  ETSI NFVや3GPPなどの国際標準化を推進し，
  将来のネットワークインフラの発展に貢献します


HERE COMES
5G
JAPAN 2020