機械学習を活用した仮想5Gコア網におけるネットワーク障害解析の取り組み

KDDI/KDDI総合研究所
河崎 純一

2022年1月27日

謝辞：本研究開発は、総務省の「革新的AIネットワーク統合基盤技術の研究開発（JPMI00316）」によって実施した成果を含みます。

ネットワークにおける仮想化の進展

NW Architectureの推移

専用物理機器上に
個別のNW機能を実装

PNF

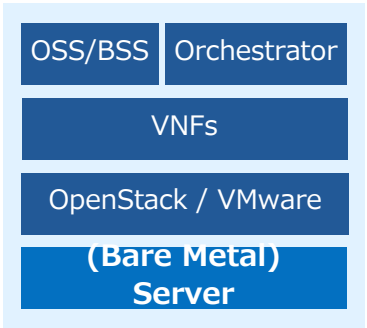
汎用物理サーバ上に仮想化
されたNW機能を実装

VNF

クラウド環境上にクラウド
ネイティブ化されたNW機能を実装

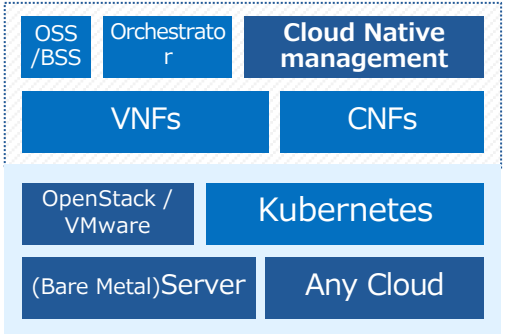
CNF

インフラレイヤの効率化　→　NW機能の仮想化



仮想レイヤの導入により、インフラレイヤの効率的な利用を目的とする。管理対象の増加やカーネルレイヤでの複雑性が上昇。オーケストレータを利用した複雑性への対処が一般的。

インフラレイヤの抽象化→NW機能のクラウドネイティブ化



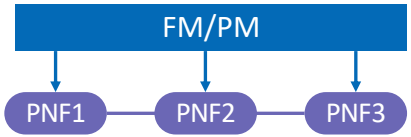
クラウドネイティブ管理技術の活用により、インフラレイヤの抽象を行う事で、NW機能のより効率的なインフラレイヤ利用が検討され始めている。

仮想化による監視対象の増加

■ 3ノードの場合の比較

- 物理環境のみの場合は3ノードの状態監視で良い
- 仮想環境をはさむと監視対象は爆発的に増加
- 下記例では監視対象は8に増加

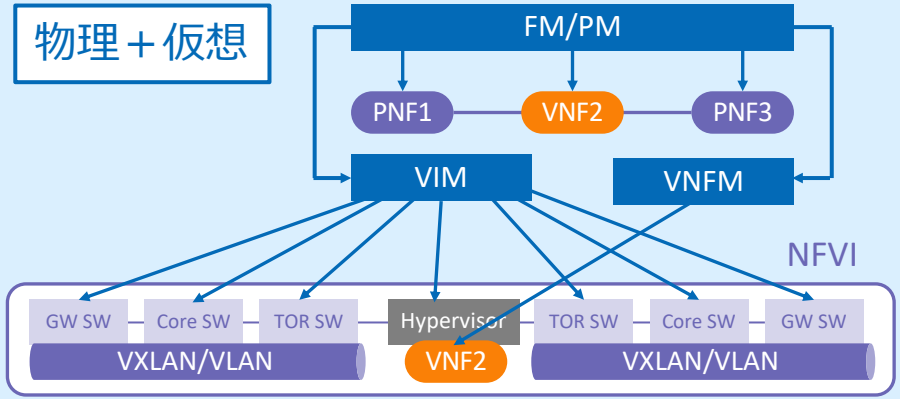
物理環境のみ



Seq	Check
1	PNF1 Device State Check : OK
2	PNF2 Device State Check : NG
3	PNF3 Device State Check : OK

Monitoring Items
3 → 8

物理＋仮想



Seq	Check
1	PNF1 Device State Check : OK
2	VNF2 Device State Check : NG
3	PNF3 Device State Check : OK
4	NFVI Hypervisor Check : NG
5	NFVI Network GW SW Check : OK
6	PNFVI Network Core SW Check : OK
7	NFVI Network TOR SW Check : OK
8	NFVI Network VXLAN/VLAN Check : OK

機械学習の導入

仮想化によって構成要素が増えることで、

AIにとっては
データが増えるのは
いいこと

運用が大変に

機械学習を使った
運用で負荷軽減！

運用プロセス毎の適用例

運用プロセス	機能要求	機械学習モデル例	pros	cons
障害検知	障害発生時に、その異常を検出し、アラートを出力する	教師なし・回帰モデル（AutoEncoder等）	・正常時のデータから構築可能	・異常だけでは適切な復旧対処の判断が難しい
障害原因解析	障害発生時に、障害解析を行い、その原因（どこで・何が）を出力する	教師あり・分類モデル（DNN等）	・出力情報から適切な復旧対処を判断可能	・十分な障害データが必要
復旧手順作成	障害を復旧させるための手順を作成する	強化学習（DQN等）	・人では思いつかない良い手順を導出できる可能性がある	・手順の妥当性の説明が難しい

本日の発表では、機械学習を活用した障害原因解析についてお話しします

本日お話しすること

- AI評価システムの構成・仕組み
  - データ生成
  - データ収集
  - データ前処理
- AIの比較評価
  - アルゴリズム
  - データセット量
  - データ項目
  - 差分処理

教師データどうする？

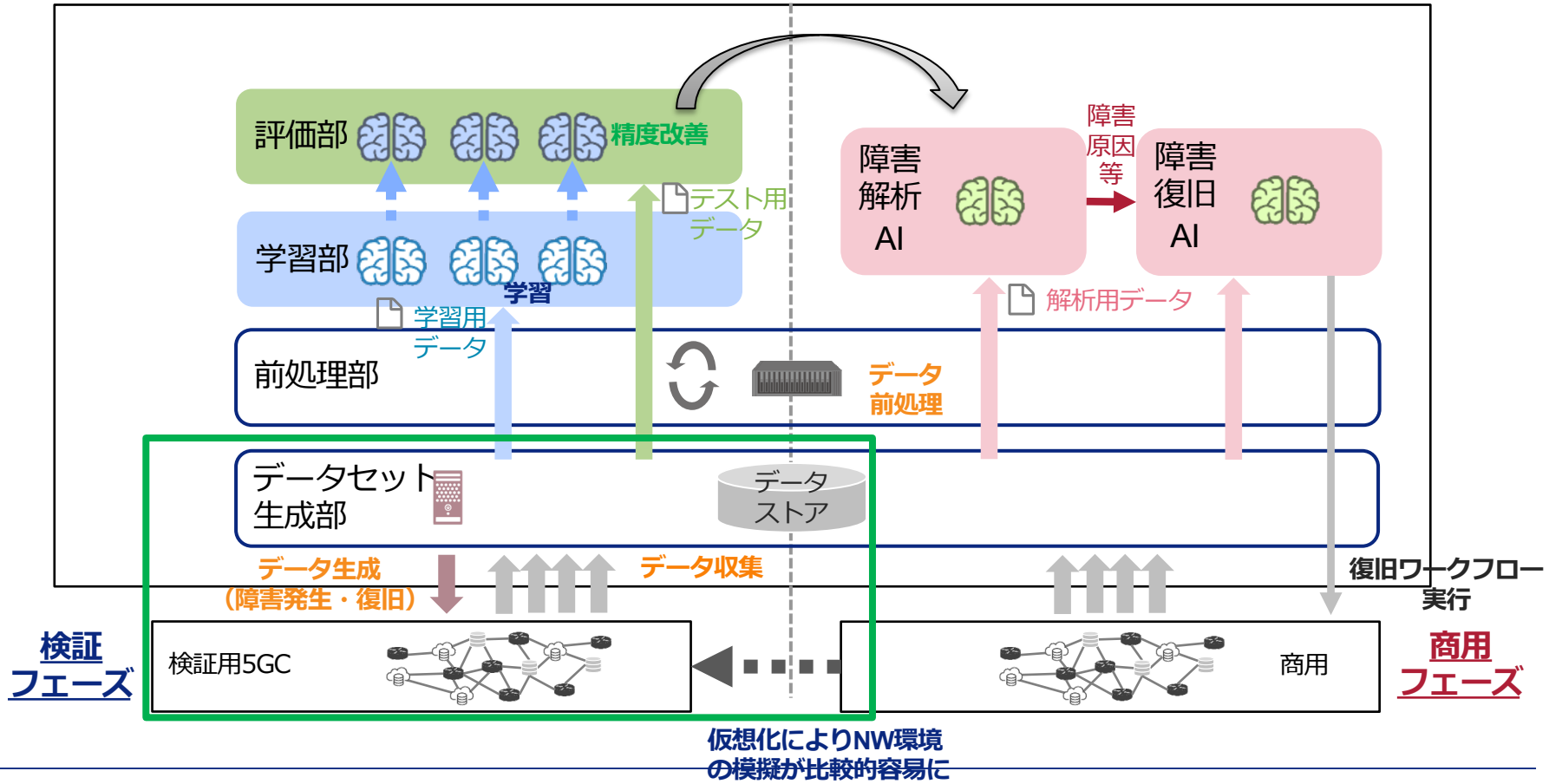
- 商用のネットワークからデータを持ってくる？
  - 障害データの数・種類が少ない
  - 監視データの収集間隔が長い

- オープンデータ？？
  - 障害種別が限定される（セキュリティ攻撃、BGP経路障害等）
  - 監視項目が限定される

やりたい検証を行うためには自分達でデータを作るしかない

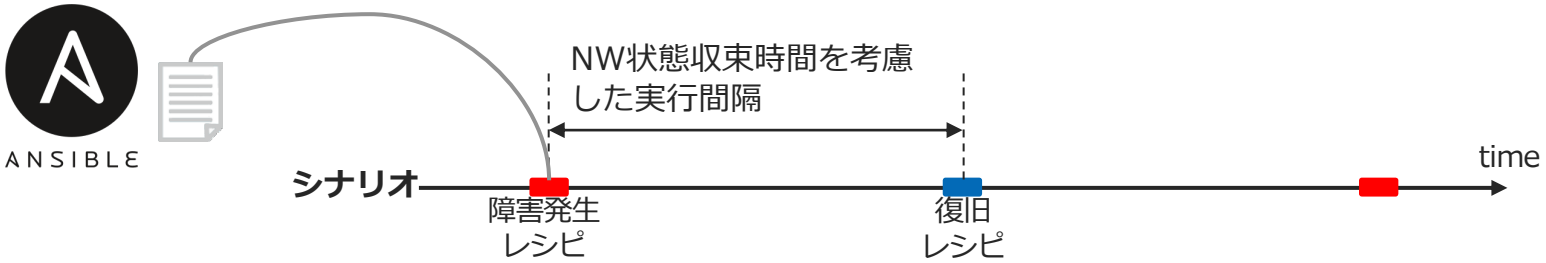
⇒　データ生成から（仮想5Gコア網の構築から）始めてみる

システム構成

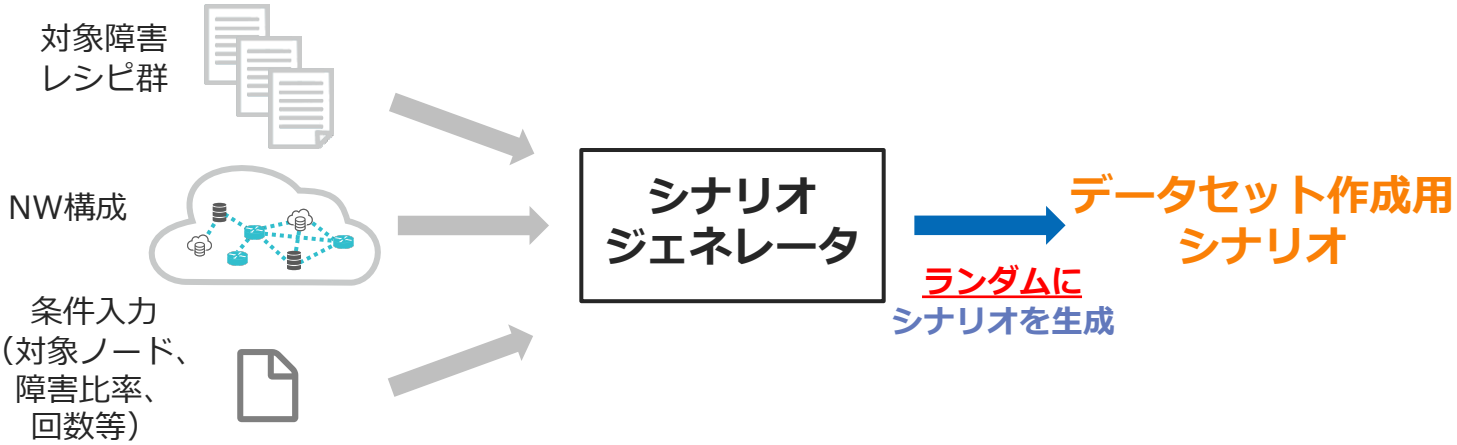


データ生成（障害発生・復旧）

障害発生/復旧レシピを組み合わせたシナリオを自動で実行することにより
必要十分なデータセットを効率的に生成



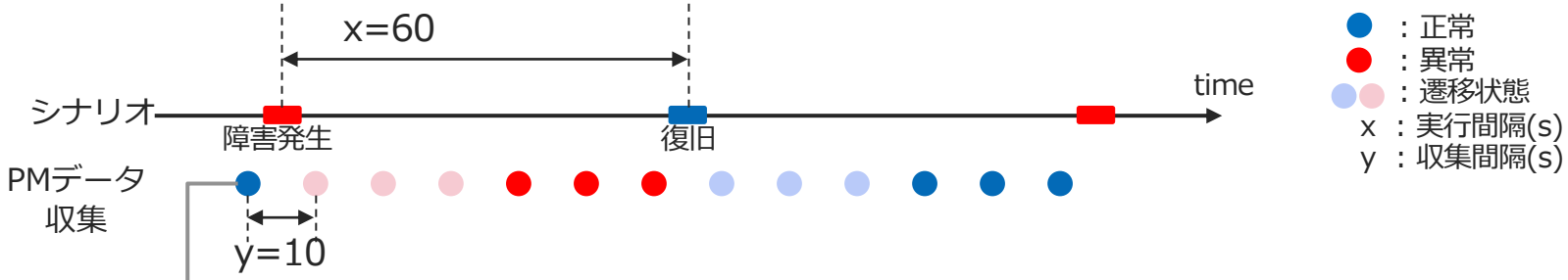
シナリオ生成方法



シナリオ例

time	type	project	node	network	port	ifname	compute	sp	original_sp	status	started_at	stopped_at
180	memory-stress-start	admin	ausfc1							succeeded	2021-02-22T16:06:50	2021-02-22T16:06:59
60	memory-stress-stop	admin	ausfc1							succeeded	2021-02-22T16:07:59	2021-02-22T16:08:06
180	vcpu-overload-start	admin	ausfb1							succeeded	2021-02-22T16:11:06	2021-02-22T16:11:15
60	vcpu-overload-stop	admin	ausfb1							succeeded	2021-02-22T16:12:15	2021-02-22T16:12:24
180	interface-loss-start	admin	udmc1			ens4				succeeded	2021-02-22T16:15:24	2021-02-22T16:15:31
60	interface-loss-stop	admin	udmc1			ens4				succeeded	2021-02-22T16:16:32	2021-02-22T16:16:39
180	interface-down	admin	ausfc1			ens4				succeeded	2021-02-22T16:19:39	2021-02-22T16:19:47
60	interface-up	admin	ausfc1			ens4				succeeded	2021-02-22T16:20:47	2021-02-22T16:20:55
180	bridge-delif	admin	amfc1	nw-cpc1						succeeded	2021-02-22T16:23:55	2021-02-22T16:24:06
60	bridge-addif	admin	amfc1	nw-cpc1						succeeded	2021-02-22T16:25:06	2021-02-22T16:25:16
180	vcpu-overload-start	admin	ausfc1							succeeded	2021-02-22T16:28:16	2021-02-22T16:28:25
60	vcpu-overload-stop	admin	ausfc1							succeeded	2021-02-22T16:29:25	2021-02-22T16:29:33
180	memory-stress-start	admin	amfb1							succeeded	2021-02-22T16:32:33	2021-02-22T16:32:42
60	memory-stress-stop	admin	amfb1							succeeded	2021-02-22T16:33:42	2021-02-22T16:33:50
180	interface-loss-start	admin	amfb1			ens5				succeeded	2021-02-22T16:36:50	2021-02-22T16:36:58
60	interface-loss-stop	admin	amfb1			ens5				succeeded	2021-02-22T16:37:59	2021-02-22T16:38:07
180	interface-down	admin	amfa1			ens5				succeeded	2021-02-22T16:41:07	2021-02-22T16:41:14
60	interface-up	admin	amfa1			ens5				succeeded	2021-02-22T16:42:15	2021-02-22T16:42:22
180	bridge-delif	admin	udmb1	nw-cpb1						succeeded	2021-02-22T16:45:22	2021-02-22T16:45:33
60	bridge-addif	admin	udmb1	nw-cpb1						succeeded	2021-02-22T16:46:33	2021-02-22T16:46:47
180	interface-loss-start	admin	ausfa1			ens4				succeeded	2021-02-22T16:49:47	2021-02-22T16:49:55
60	interface-loss-stop	admin	ausfa1			ens4				succeeded	2021-02-22T16:50:55	2021-02-22T16:51:02
180	bridge-delif	admin	amfb1	nw-cpb1						succeeded	2021-02-22T16:54:03	2021-02-22T16:54:12
60	bridge-addif	admin	amfb1	nw-cpb1						succeeded	2021-02-22T16:55:13	2021-02-22T16:55:22
180	memory-stress-start	admin	amfc1							succeeded	2021-02-22T16:58:22	2021-02-22T16:58:31
60	memory-stress-stop	admin	amfc1							succeeded	2021-02-22T16:59:31	2021-02-22T16:59:40
180	interface-loss-start	admin	amfa1			ens5				succeeded	2021-02-22T17:02:40	2021-02-22T17:02:50
60	interface-loss-stop	admin	amfa1			ens5				succeeded	2021-02-22T17:03:51	2021-02-22T17:03:58
180	memory-stress-start	admin	ausfa1							succeeded	2021-02-22T17:06:58	2021-02-22T17:07:09
60	memory-stress-stop	admin	ausfa1							succeeded	2021-02-22T17:08:09	2021-02-22T17:08:18
180	interface-loss-start	admin	amfc1			ens5				succeeded	2021-02-22T17:11:18	2021-02-22T17:11:28
60	interface-loss-stop	admin	amfc1			ens5				succeeded	2021-02-22T17:12:29	2021-02-22T17:12:36

データ収集



ZABBIX

Zabbixエージェント/CLIにより各ノードから収集されるデータ

- ノード
  - デバイス
    - software-oper（CPU負荷情報等）
    - five-g（セッション数情報等）
  - インタフェース
    - interfaces（トラフィック情報等）

収集データ例

収集時間

ノード名

インタフェース情報

```
{
  "_index": "network-5gc-20201118",
"_id": "2020-11-18T15:24:30+0900",
  "_version": 1,
  "_score": null,
  "_source": {
~~~~~~
    "smf": [
     {
      "name": "smf",
       "interfaces": {
~~~~~~
          {
           "name": "ens4",
           "type": "ianaift:other",
           "admin-status": "up",
           "oper-status": "up",
           "last-change": null,
           "if-index": 3,
           "phys-address": "fa:16:3e:d9:61:4b",
           "speed": -1,
           "statistics": {
            "discontinuity-time": null,
            "in-octets": 7158624101,
            "in-unicast-pkts": 25401825,
            "in-broadcast-pkts": 0,
            "in-multicast-pkts": 0,
            "in-discards": 0,
            "in-errors": 0,
            "in-unknown-protos": 0,
            "out-octets": 4433573212,
            "out-unicast-pkts": 25388727,
            "out-broadcast-pkts": 0,
            "out-multicast-pkts": 0,
            "out-discards": 0,
            "out-errors": 0
          }
```

```
"memory-stats": {
                "memory-status": "Healthy",
                "total": 2047856,
                "used-number": 812004,
                "used-percent": 40,
                "free-number": 1235852,
                "free-percent": 60,
                "available-number": 1509312,
                "available-percent": 74,
                "committed-number": 1278364,
                "committed-percent": 62,
                "status": {
                 "warning-threshold-percent": 88,
                 "critical-threshold-percent": 93
                }
               },
               "per-core-stats": {
                "per-core-stat": [
                 {
                  "name": 0,
                  "user": 2,
                  "system": 4,
                  "nice": 0,
                  "idle": 94,
                  "irq": 0,
                  "sirq": 0,
                  "io-wait": 0
                 },
~~~~~
```

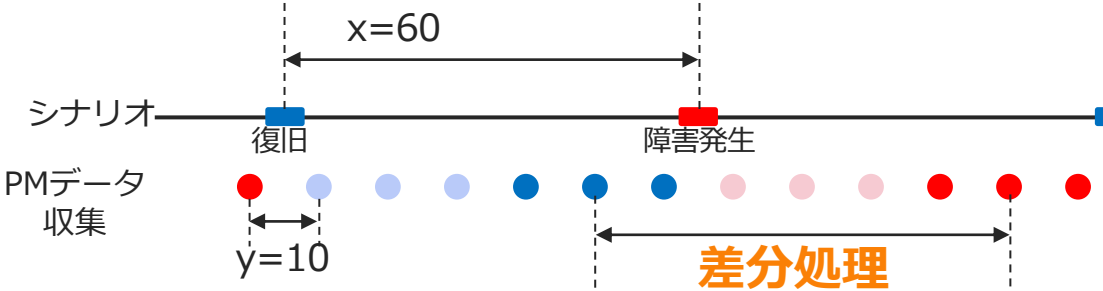
メモリ・CPU情報

```
"five-g": {
       "SM": {
        "SessionNbrMean": 350,
        "SessionNbrMax": 350,
        "PduSessionCreationReqNSI": 225307,
        "PduSessionCreationSuccNSI": 225305
       }
      }
     }
    ],
~~~~~
    "time": 1605680670,
    "@timestamp": "2020-11-18T15:24:30+0900",
    "@log_name": "network-5gc"
  },
  "fields": {
   "@timestamp": [
    "2020-11-18T06:24:30.000Z"
   ]
  },
  "sort": [
   1605680670000
  ]
}
```

5G情報

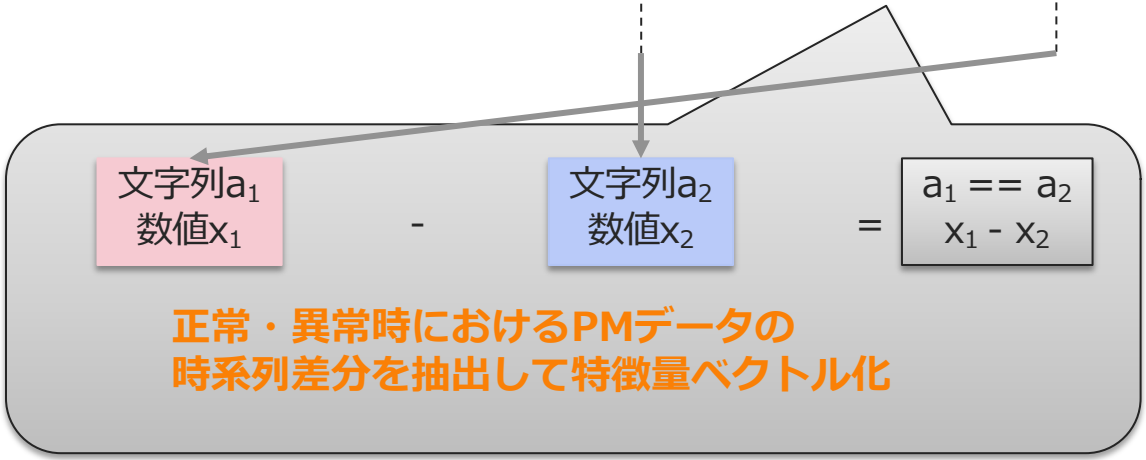
データ前処理その１

障害前後のPMデータの差分を特徴量として抽出することによって、
障害の変化点を捉えたデータセットに変換



● ：正常
● ：異常
● ● ：遷移状態
x ：実行間隔(s)
y ：収集間隔(s)

差分処理



正常・異常時におけるPMデータの
時系列差分を抽出して特徴量ベクトル化

例）PDUセッション確立成功数

前処理前）
障害発生前　：x_2=225305
後　：x_1=225310

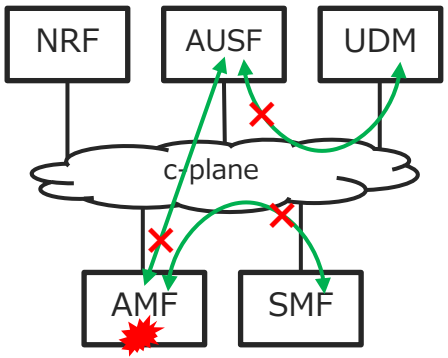
前処理後）
差分抽出無　→　225310
有　→　5

解析上意味のあるデータに変換

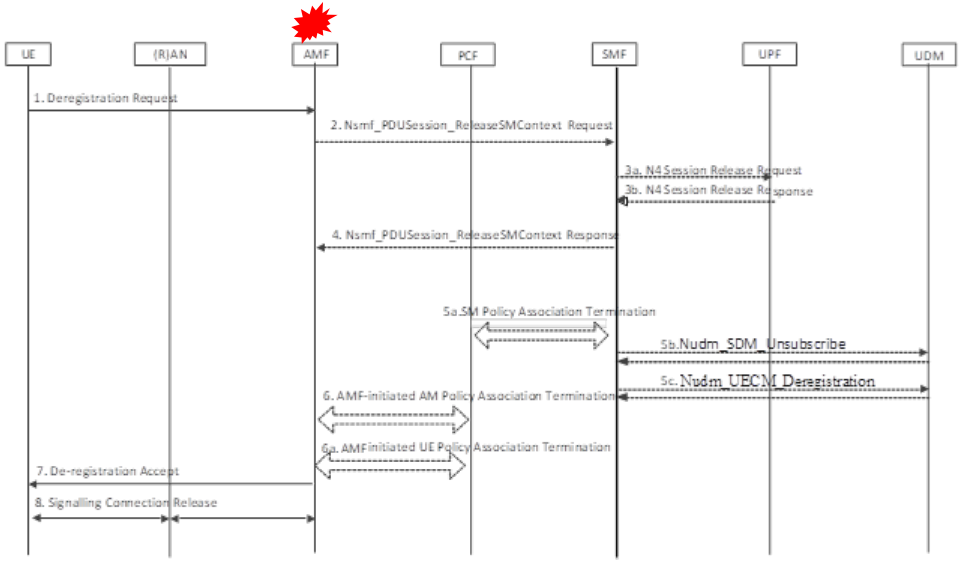
データ前処理その２

5Gコアの場合、3GPPの規定に従って各ノードがシーケンシャルに処理を実行

・5Gコア（Cプレーン）



・UE-initiated Deregistration

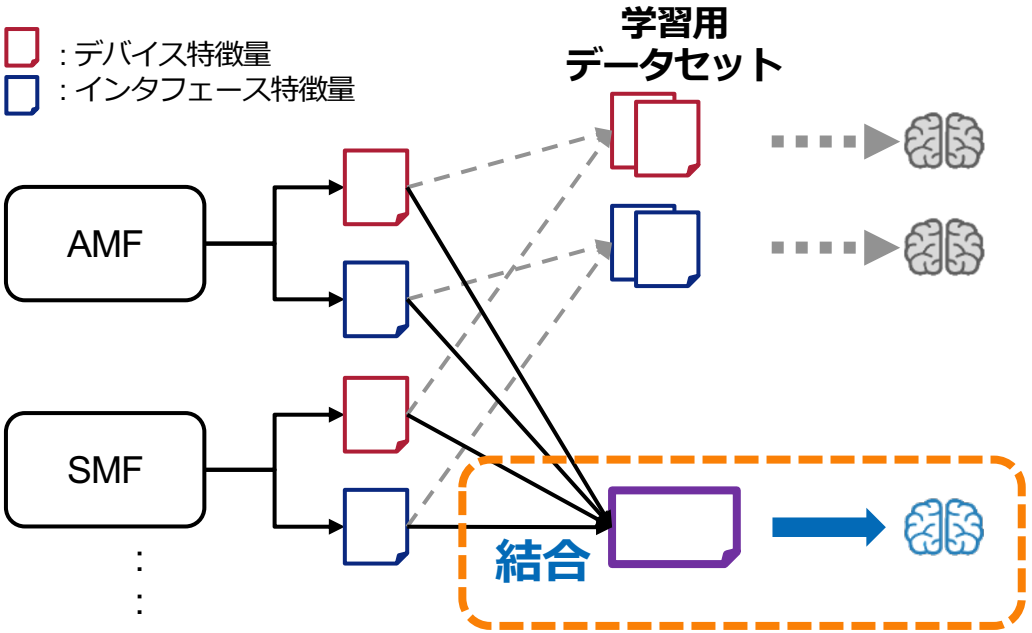


Ref. 3GPP TS 23.502

あるノード障害が他ノードにも波及するため、関連ノードのデータを集約して分析

データ前処理その２

分析精度向上のために関連するノードの特徴量を結合



関連ノードから得られる全特徴量をマージすることで、NW単位の解析を可能に

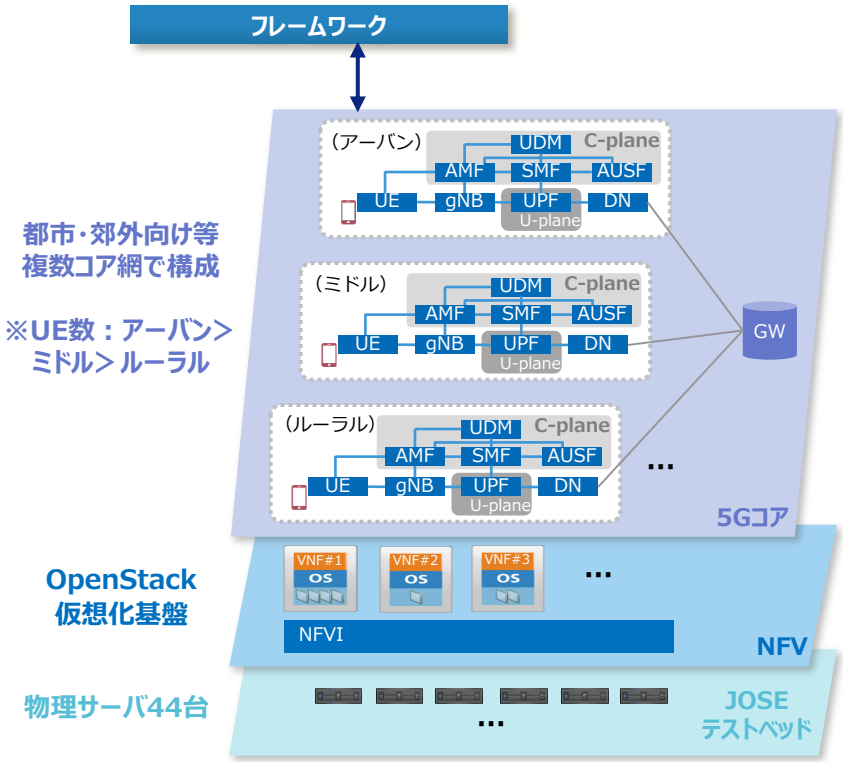
AI評価

評価システムを用いて多種多様な教師データを作り、障害原因解析モデルの評価を行った

対象ネットワーク：VNFベースの仮想5Gコア網
モデルの予測対象：障害発生時にどこで・何が起きたか

AI評価

評価環境（大規模5Gモバイルコア網）



対象障害

- インタフェース断
- 仮想ブリッジ断
- パケットロス
- メモリ高負荷
- CPU高負荷

※シナリオ条件

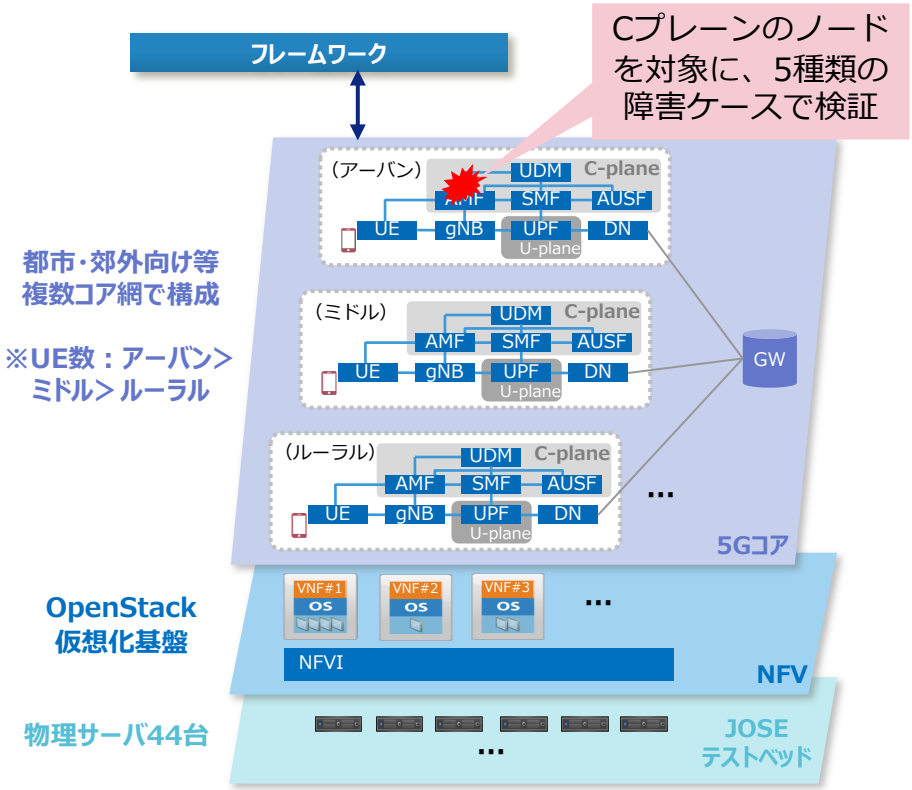
対象ノード：AMF, UDM, AUSF
障害比率：均一
障害発生回数：各ノード同じ

シナリオ例

Seq	レシピ	エリア	ノード
1	パケットロス	ルーラル	SMF
2	復旧		
3	インタフェース断	アーバン	AUSF
4	復旧		
5	メモリ高負荷	ミドル	AMF
6	復旧		
	:		

AI評価

● 評価環境（大規模5Gモバイルコア網）



● 評価方法

ランダムシナリオにより生成されたデータセットを用いて学習したモデルを、異なる時間帯に別のランダムシナリオで作られたデータセットを用いて評価

- **データセット詳細**

データセット	正常	インタフェース断	仮想ブリッジ断	パケットロス	メモリ高負荷	CPU高負荷
学習用	7370	714	712	713	711	715
評価用	1764	171	171	171	171	171

- **評価軸**
  - アルゴリズム
  - データセット量
  - データ項目
  - 差分処理

アルゴリズム比較

- **評価条件**

学習用データセット：全て利用
評価用データセット：全て利用
アルゴリズム：3種

- **結果**

アルゴリズム		Cプレーン IF障害			高負荷 障害	
		IF断	ブリッジ断	パケロス	メモリ	CPU
MLP	Precision	100%	82.3%	41.7%	99.4%	94.2%
	Recall	98.8%	33.4%	29.2%	100%	100%
RandomForest	Precision	98.3%	92.4%	95.4%	100%	100%
	Recall	100%	43.3 %	67.8 %	100%	100%
SVM	Precision	0%	0%	0%	0%	0.7%
	Recall	0%	0%	0%	0%	33.3%

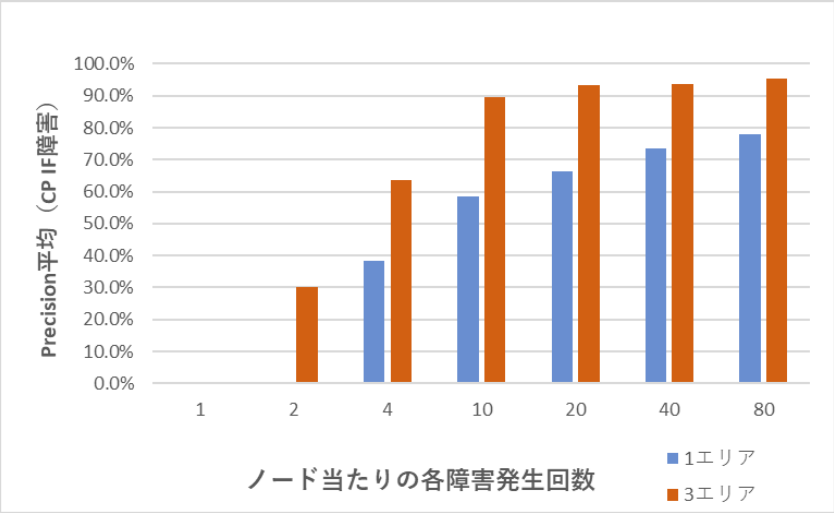
RandomForestが高精度

データセット量比較

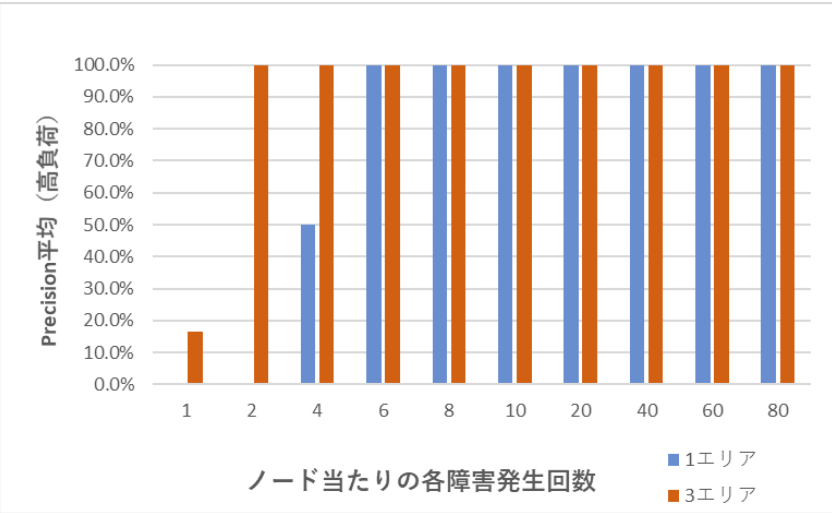
- **評価条件**

学習用データセット：ノード当たりの各障害発生回数（1/2/4/10/20/40/80回）＊
全3エリア/1エリア（アーバン）を利用
評価用データセット：全て利用
アルゴリズム：RandomForest

- **CP IF障害**



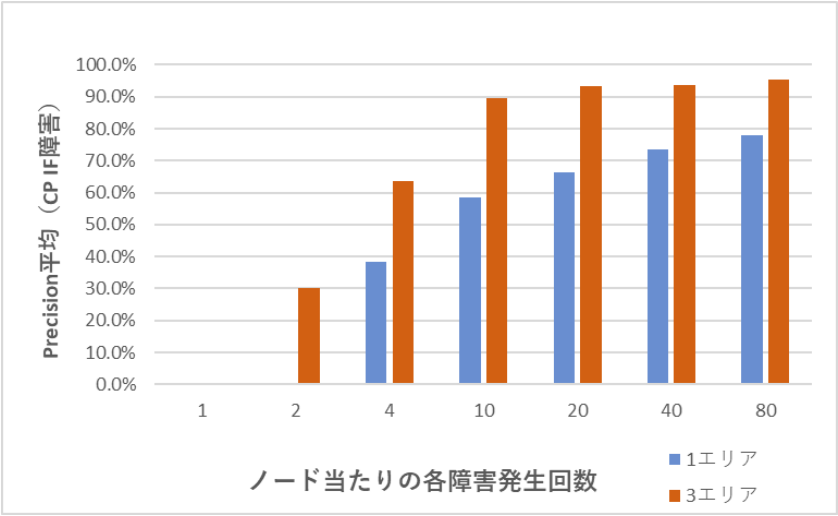
- **高負荷障害**



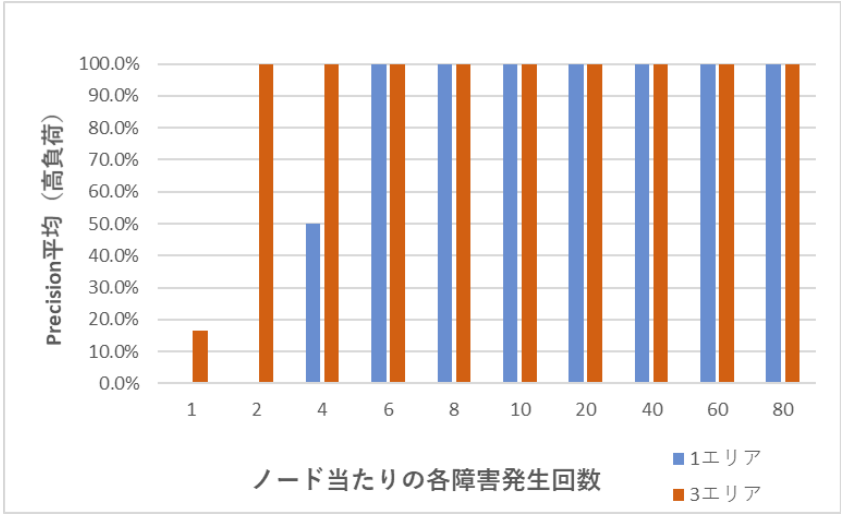
データセット量比較

- ✓ 各ノード1回ずつの障害データでは、有効なモデルを作るのに十分でない
- ✓ 高い精度のモデルを得るために必要なデータセット量は障害によって異なる（Precision90%以上の場合、CP IFの場合20回以上、高負荷の場合2回以上）
- ✓ ある特性の1エリア（UE数の多いアーバン）のみで収集したデータだけで学習すると、全エリアのデータを用いる場合に比べて精度が上がりにくい（特にCP IFの場合）

- **CP IF障害**



- **高負荷障害**



データ項目・差分処理有無

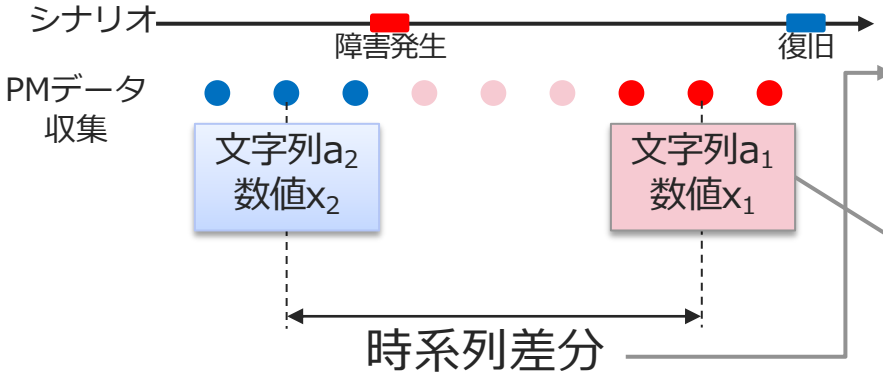
- **データ項目**

学習用データセット：特徴量（全て、メモリ/CPU情報除く、5G情報のみ）を利用
評価用データセット：全て利用
アルゴリズム：RandomForest

データ収集項目	Precision平均	
	CP IF	高負荷
全て	95.4%	100%
メモリ/CPU除く	92.5%	66.9%
5Gのみ	12.1%	1.9%

障害種別によって必要な項目が不足していると精度が低下

- **差分処理有無**



Precision平均	
CP IF	高負荷
95.4%	100%

Precision平均	
CP IF	高負荷
12.6%	66.7%

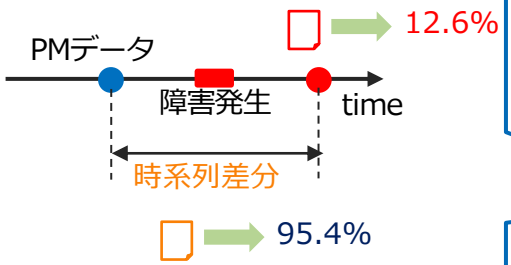
差分処理無の場合、精度低下

評価まとめ

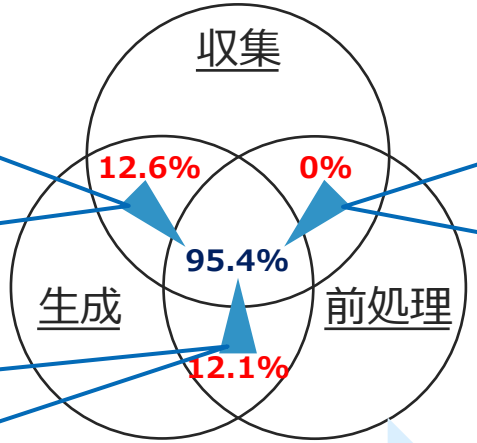
- **AI作成フロー**　**学習前のプロセス**も精度に大きく影響

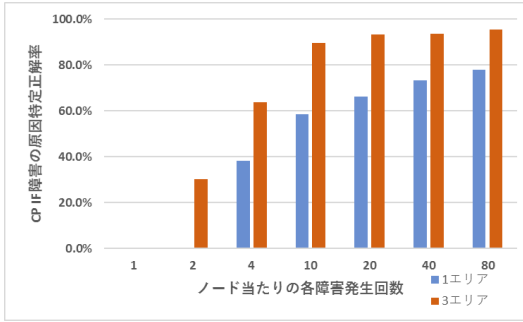
(1)データ生成 → (2)データ収集 → (3)データ前処理 → (4)学習

- 差分処理有無による精度差



各プロセスの処理差分による
CP IF障害のPrecision平均





- 障害発生回数と精度の関係

データ収集項目	精度
モバイル固有のみ	12.1%
インフラ含む	95.4%

商用でのAI適用には、学習だけでなく、
その前のプロセス含めた**全体設計**が必要

まとめと議論ポイント

■ 本日のまとめ

将来ネットワークの運用を支えうる機械学習の可能性を探るべく、

- ✓ 教師ありの障害原因分析モデルの構築手法
- ✓ 仮想5Gコア網を対象とした各種比較実験

についてお話ししました

■ 皆様に特に聞きたいこと

- 他に機械学習が使えそうな障害ケースや運用業務について
- 何かしら機械学習を商用導入or検討された方がいれば、課題や解決策について
- 他にも、何でもコメント・質問お待ちしております


Tomorrow, Together
KDDI
おもしろいほうの未来へ。
au