# VNF/CNF基盤におけるCCNベーステレメトリ機構の試作

**平山 孝弘**
国立研究開発法人 情報通信研究機構
ネットワーク研究所
ネットワークアーキテクチャ研究室

Takahiro Hirayama [ hirayama@nict.go.jp ]

# あらまし

- サービスの多様化、制御対象の増加(端末数の増加やVNF/CNFの細分化)、広域化(TN/NTN)など、ネットワーク基盤の管理制御の複雑さが増している
  →**管理制御の自動化**

- 主な取り組み
  - 需要予測に基づく、リソース調停や機能マイグレーションを行うAI
  - 複数のAIを連携動作させるためのIF開発、キャリア・ベンダとの共同実証
  - Intent Based Networking (IBN)
    - 入力するのは要求だけ。設計、設定、運用など必要な処理はAIが行う
  - **AIがネットワーク基盤内の稼働状況を正確に把握するためのテレメトリ技術**
    - CCNベースのテレメトリ情報共有基盤の研究開発

# アウトライン

- **これまでの取り組み**
  - **VNF/CNF基盤の需要予測、移行スケジューリング決定**
  - **AI間連携IFを用いたサービス生成・障害予測・退避処理の自動化の共同実証**

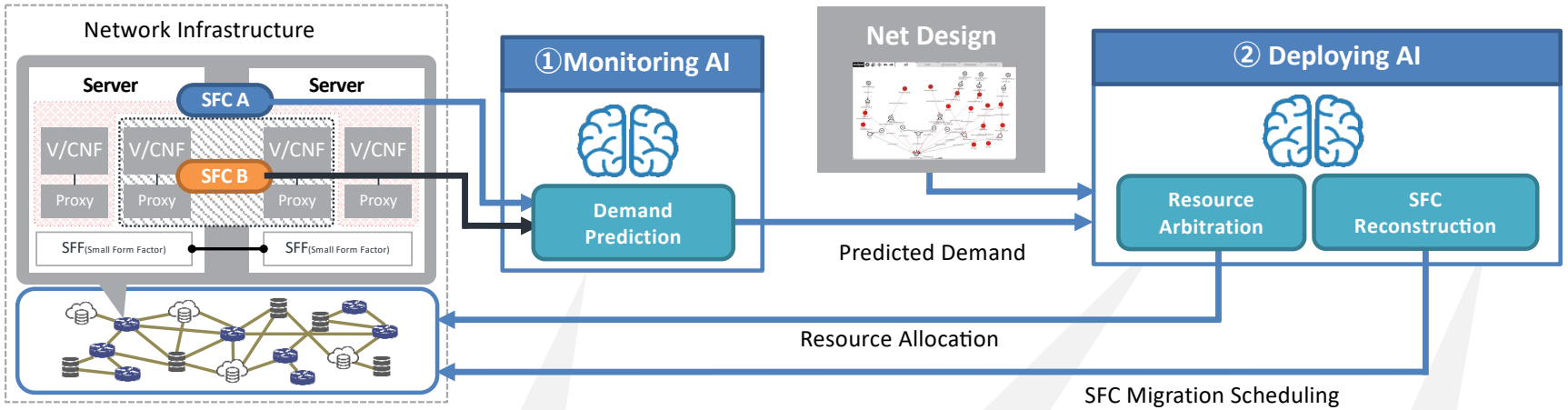
- CCNベースのテレメトリ情報共有機構

- LLMによる情報取得機能の試作

# Autonomic Resource Management in SFC Platform

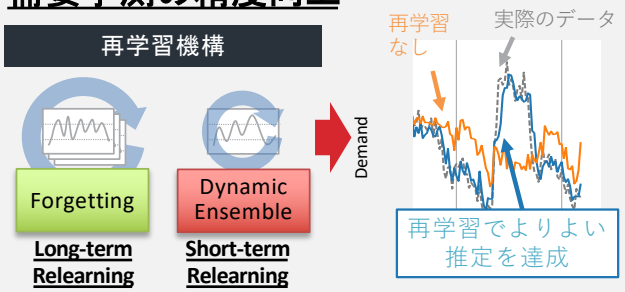
**AI活用による自動自律制御により、サービスネットワーク基盤を対象に、秒単位の資源調停と、機能移行時間の最小化を図りつつ、サービス品質を維持**

KVM docker 環境



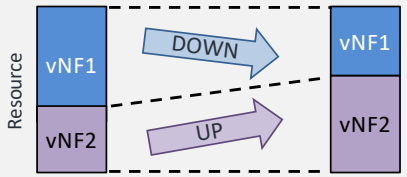

To Operating Phase

## 需要予測の精度向上



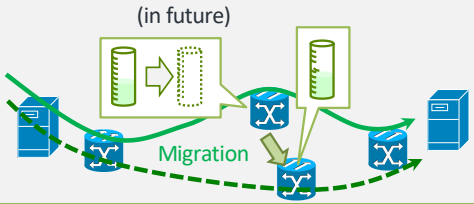

Cf. T. Hirayama *et al.*, IEEE NetSoft'20.
T. Hirayama *et al.*, IEICE Trans. Inf. and Sys., '21.

## 同一サーバ内のVNF間でリソースを自動調停




Cf. T. Miyazawa *et al.*, IEEE/IFIP NOMS'18.
T. Hirayama *et al.* iPOP'20,

## リソース不足を避けるVNFマイグレーションスケジューリングAI



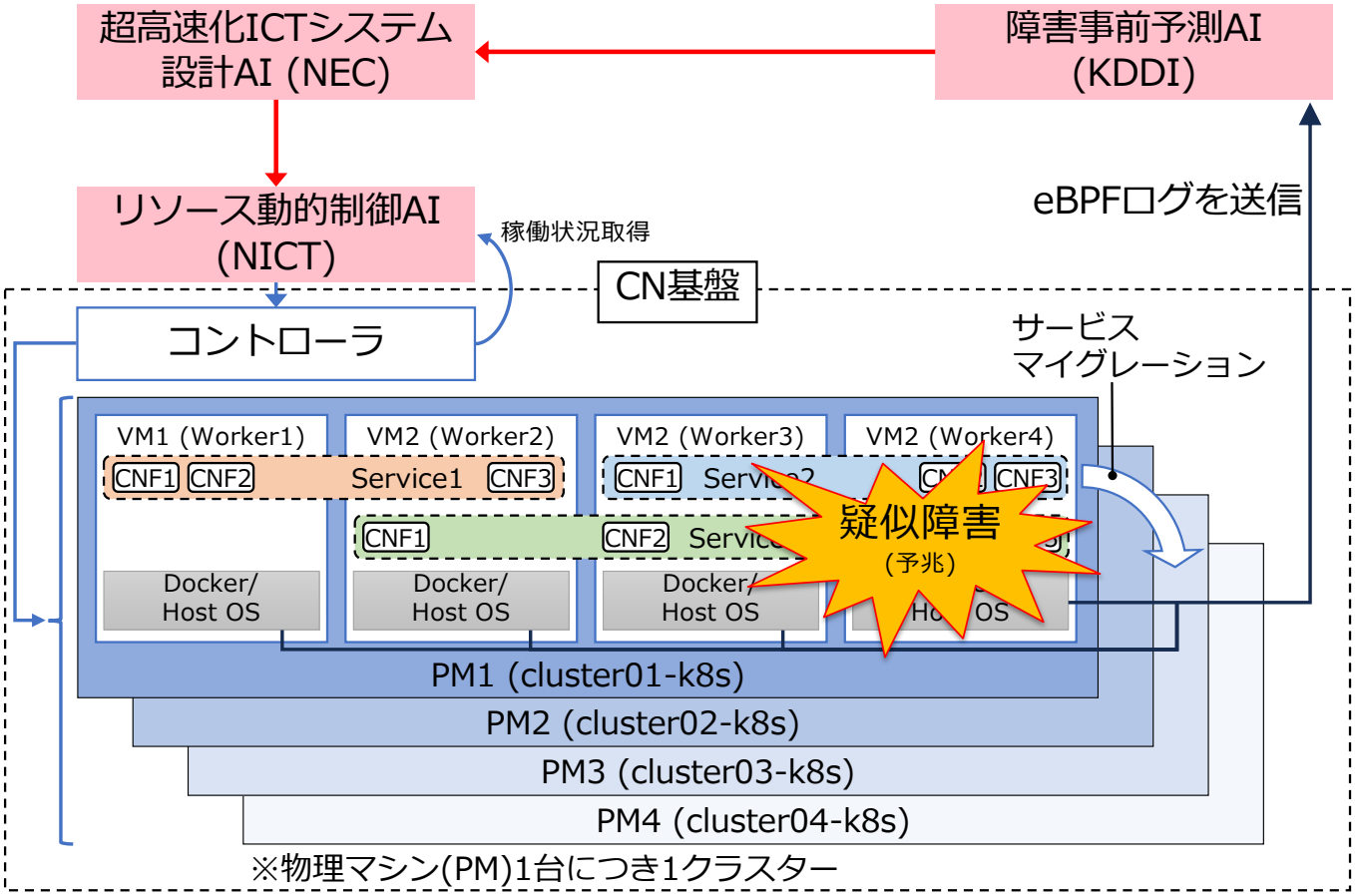

Encoder-Decoder Recurrent Neural Network

Cf. T. Hirayama *et al.*, IEEE NetSoft'19.
T. Hirayama *et al.*, IEEE CNSM'22.

# 3種のAIの連携によるサービス自動管理制御

- 目的
  - NICTテストベッド上の検証基盤で稼働するサービスをAIで自動管理
- 成果
  - 3社が開発した3種のAIを連携させ、
    - ① サービス生成
    - ② 障害予測とサービスの対比
      - a. 障害事前予測(予兆の検出)
      - b. 障害を避けたサービスの再設計(マイグレーション先クラスタの決定)
      - c. 移行先でのリソース競合回避(リソース競合を避けたCNF配置)

    の自動化を実証
  - 連携IFはTMF Open APIを改良
    - Service Problem API
    - Service Order API
    - Service Inventory



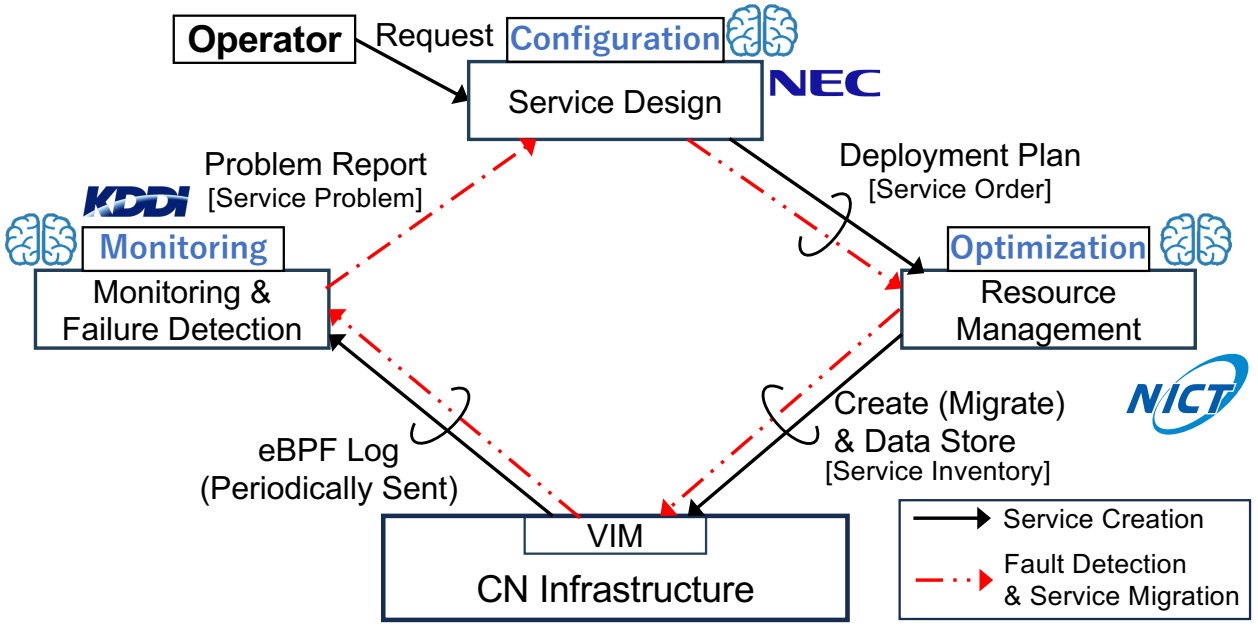

Hirayama et al., NOMS'25
APNOMS'25でTutorial予定

# Closed-loop Management architecture & Workflow

3種のAIモデルが連携しクローズドループ制御を自動化

- 2つのケースを実証

1. 既設サービスとの競合を避けた新規サービスの生成 (黒)
2. 障害の予兆を検知し、障害箇所を避けるためにサービスを再構成 (赤)



# 連携IF: TMF Open API (Service Problem API)の改良

- インフラ提供者などがアラート、QoSの劣化、SLA違反や障害発生などを通知するためのAPI
- 障害予測AI→システム設計AIへ障害の影響を受けるサービス名を通知する際に使用
  - http://{システム設計 AI の IP アドレス}/tmf-api/serviceProblemManagement/v4/ へPOSTする

- 既存のAPIでは発生した時刻を通知するもの
  →将来起こりうる障害の予告には非対応
  - 暫定的にcreationDateに発生予想時刻を記載
  - 将来的にはAPIの拡張が必要

- characteristicに障害を受けるサービス名、characteristic.valueにサービスのタイプ、障害が起きるクラスター名を記載

```
{
      "category": "system.originated",
      "priority": 1,
      "description": "Network failure will be occurred
                      at ec-site.",
      "reason": "Network failure",
      "originatorParty": {
           "role": "system",
           "id": "kddi-AI-system",
           "@referredType": "Organization"
      },
      "creationDate": <障害発生予想される(未来の)時刻>,
      "characteristic": [{
           "id": <サービスID>,
           "name": <障害の影響を受けるサービス名>,
           "valueType": "object",
           "value": {
                "type": "ns",
                "nf": "cnf",
                "clusterName": <障害が起きるクラスター名>
           }
      }, { (対象が複数ある場合は列挙) } ]
}
```

下線部: 必須フィールド、斜字体: 付加フィールド

例: KDDI AI→NEC AIへPOSTされるJSON

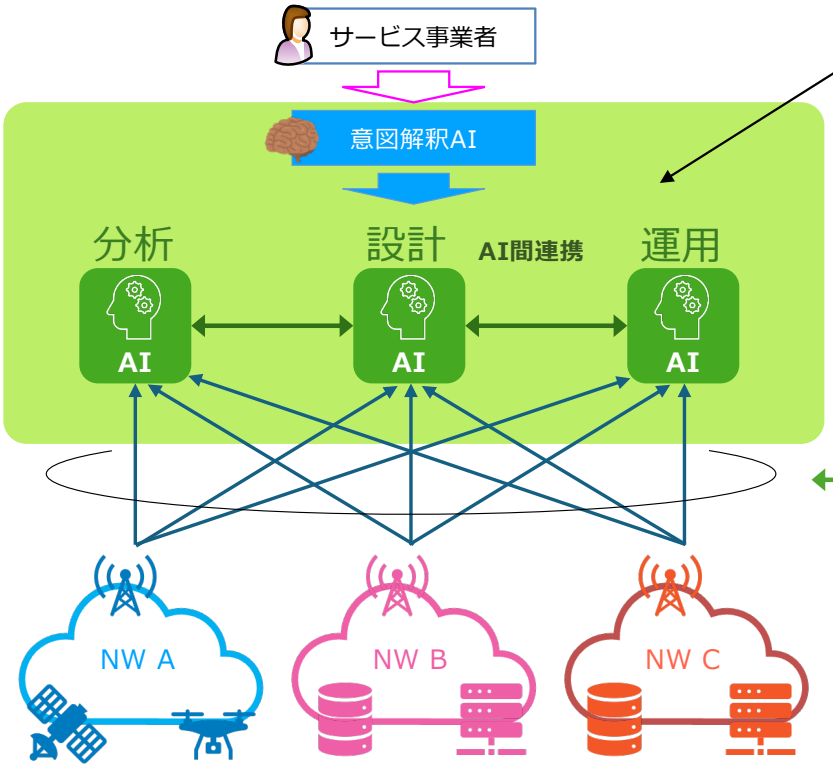
# アウトライン

- これまでの取り組み
  - VNF/CNF基盤の需要予測、移行スケジューリング決定
  - AI間連携IFを用いたサービス生成・障害予測・退避処理の自動化の共同実証

- **CCNベースのテレメトリ情報共有機構**

- LLMによる情報取得機能の試作

# 自動化への課題



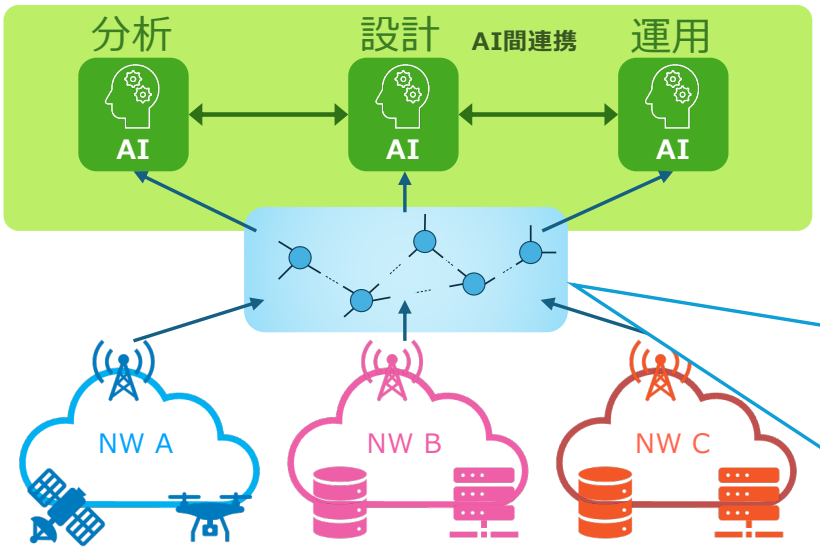

課題(IBN, AI間連携):
・要求に応じたサービスの自動構築、QoS要件の定義
**→基盤の稼働状況などの情報を正確に把握する必要**
・AI間の連携/情報伝達

本日のフォーカス

**課題: AIへ入力するテレメトリ情報の取得**
・目的(分析・設計・運用)ごとに適したモデルは異なる
→ 必要なテレメトリ情報が異なる
→ 様々なテレメトリ情報取得要求が発生
⇒ **テレメトリトラヒックの増加・複雑化**

**課題: 異なるNFV基盤*への対応**
*KVM, Kubernetes, etc.

# 開発技術の概要




**共通管理基盤(CCN)**

**負荷分散・トラヒック量削減・低遅延化**

・分散(マルチキャスト)テレメトリ情報伝送
・テレメトリ情報の圧縮、冗長な情報の削減
  - 先行研究の3倍のリクエストを処理可能*
  - 状況の変化に数十msec以内で追随*

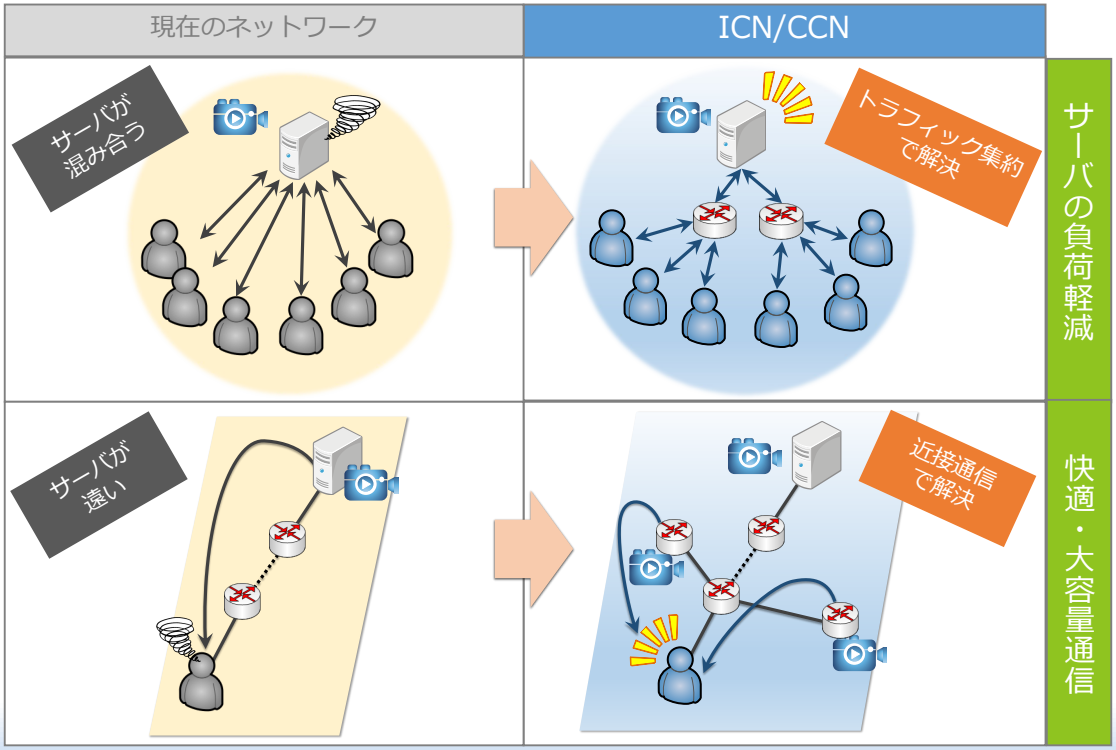
*Pedro et al, NOMS'24

**インタフェースの共通化・利便性向上**

・観測対象(サービス/VM/コンテナ etc.)や計測期間を柔軟に指定可能なインタフェース
・異なるNFV基盤のテレメトリ情報を包括的に取得

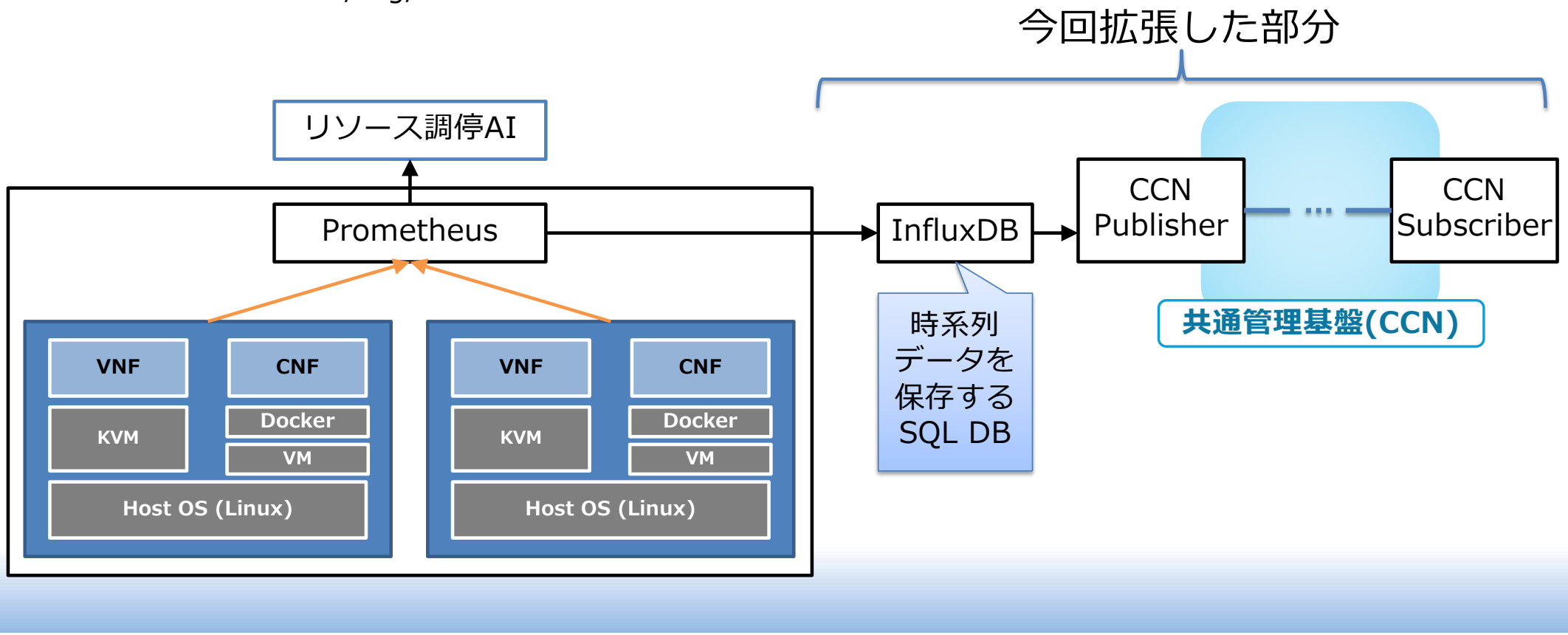
# CCN(ICN)

- 経路上のルータに情報をキャッシュし、他のクライアントにコピーを提供する
  - 1 vs. 多のサーバ-クライアント方式に比べて負荷軽減。頻出するリクエストには即応可能
- ネームベース(どんな情報が欲しいか)でのリクエストが可能



# 既設のVNF/CNF基盤との統合

- これまでに開発した情報提供機能(Prometheusベース)を拡張
- 一旦InfluxDBに蓄積し、CCNのリクエスト受信時に内容に応じて情報取得
  - CCN URI: 接頭辞”ccnx:/.../”の後にデータの種別、計測期間、観測対象、などを自由に追記可能。
    - 期間の指定、max/avg/minの取得などはInfluxDBの標準機能を使用。



# アウトライン

- これまでの取り組み
  - VNF/CNF基盤の需要予測、移行スケジューリング決定
  - AI間連携IFを用いたサービス生成・障害予測・退避処理の自動化の共同実証

- CCNベースのテレメトリ情報共有機構

- **LLMによる情報取得機能の試作**

- 取得したい情報の対象、期間、ICN(CCNx)のURI形式で柔軟に指定可能
- クラスタ単位、サービス単位、VM単位、CNF単位のCPU/メモリ/トラヒック量などの情報を取得可能。計測期間の指定、max/avg/minを選択可能

**正確なURI記述ルールを把握する必要**

**→<u>LLMを用いて自然言語→URIに変換し、入力の簡易化を図る</u>**

- 例
  - VIMが管理する計算機の2025/9/4 10:00-18:00のCPU利用率(30秒ごとの平均)
    - ccnx:/.../metric/cpu_usage/vim/node5-openvim/type/compute-node/between/20250904100000-20250904180000/ave/30s
  - サービスを構成するCNFの直近12時間のメモリ利用率(10分ごとの平均)
    - ccnx:/.../metric/memory_used/ns/ns-cnf-robotshop-201/type/cnf/latest/12h/ave/1m

# 自然言語→URIへの変換

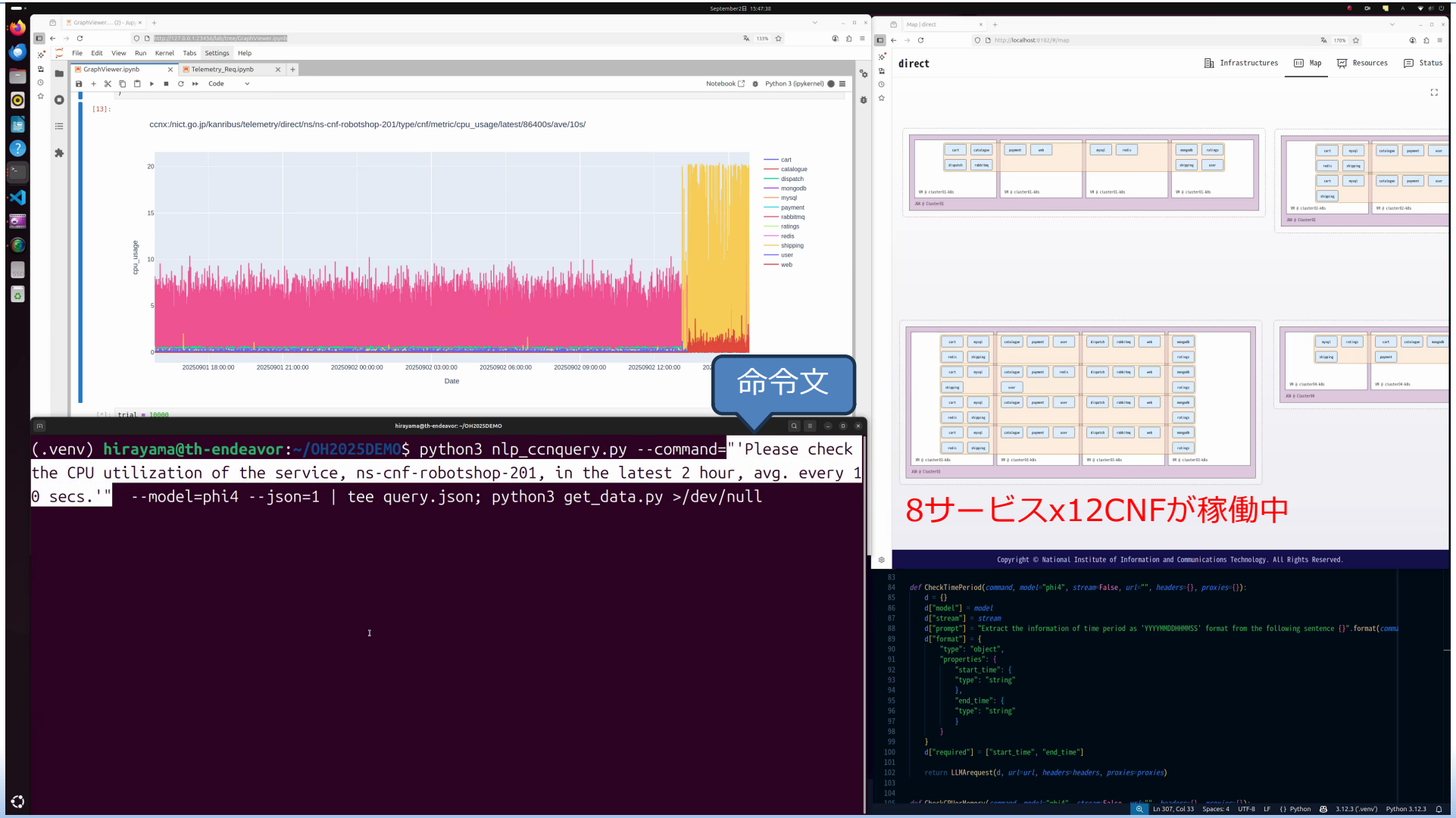
- 入力文
  - “Please check the CPU utilization of the service, ns-cnf-robotshop-201, in the latest 2 hours, avg. every 10 secs.”

- LLMに様々な質問をし、要求を分析(改良の余地あり)
  - Latestか? 日時を指定しているか?
    - Latestの場合→何秒間か確認　日時を指定している場合→日時を”YYYYMMDDHHMMSS”形式で出力
  - CPUかメモリか?
  - 固有名詞(上記の例ではサービス名)を抜き出す
  - 平均値(何秒ごと)か?

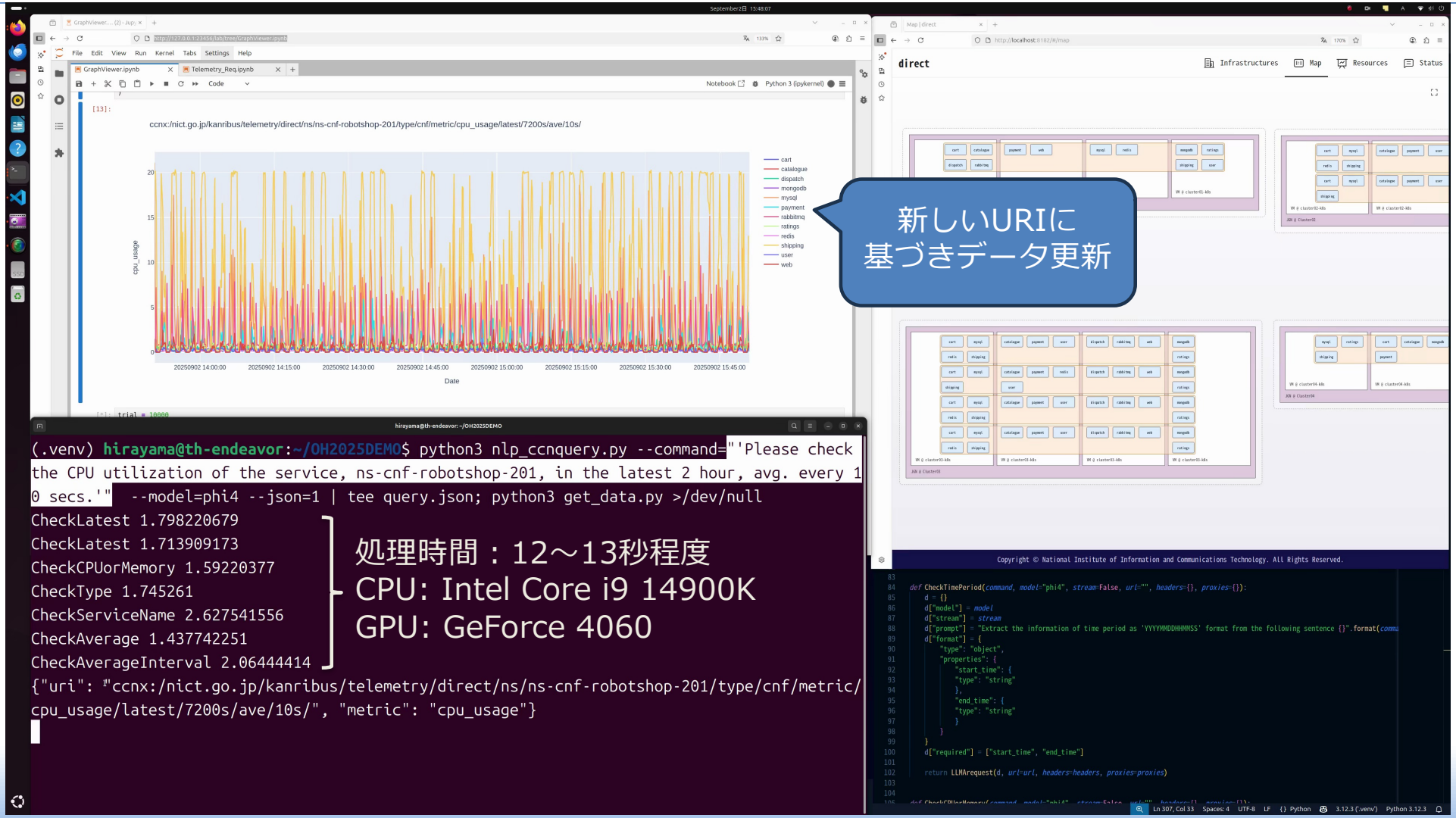
  ※何度も質問を繰り返しているため、処理時間が課題

- 戻り値(主にTrue or False)をもとにURIを生成する。
  - ccnx://.../ns/ns-cnf-robotshop201/type/cnf/metric/cpu_usage/latest/7200s/ave/10s/

# 動作例 (1/2)


ccnx:/nict.go.jp/kanribus/telemetry/direct/ns/ns-cnf-robotshop-201/type/cnf/metric/cpu_usage/latest/86400s/ave/10s/
cpu_usage
Date
cart
catalogue
dispatch
mongodb
mysql
payment
rabbitmq
ratings
redis
shipping
user
web
命令文
(.venv) hirayama@th-endeavor:~/OH2025DEMO$ python3 nlp_ccnquery.py --command="'Please check the CPU utilization of the service, ns-cnf-robotshop-201, in the latest 2 hour, avg. every 10 secs.'" --model=phi4 --json=1 | tee query.json; python3 get_data.py >/dev/null
direct
Infrastructures
Map
Resources
Status
8サービスx12CNFが稼働中
Copyright © National Institute of Information and Communications Technology. All Rights Reserved.

# 動作例 (2/2)


ccnx:/nict.go.jp/kanribus/telemetry/direct/ns/ns-cnf-robotshop-201/type/cnf/metric/cpu_usage/latest/7200s/ave/10s/
cpu_usage
Date
cart
catalogue
dispatch
mongodb
mysql
payment
rabbitmq
ratings
redis
shipping
user
web
direct
Infrastructures
Map
Resources
Status
新しいURIに
基づきデータ更新
(.venv) hirayama@th-endeavor:~/OH2025DEMO$ python3 nlp_ccnquery.py --command="'Please check the CPU utilization of the service, ns-cnf-robotshop-201, in the latest 2 hour, avg. every 10 secs.'" --model=phi4 --json=1 | tee query.json; python3 get_data.py >/dev/null
CheckLatest 1.798220679
CheckLatest 1.713909173
CheckCPUorMemory 1.59220377
CheckType 1.745261
CheckServiceName 2.627541556
CheckAverage 1.437742251
CheckAverageInterval 2.06444414
{"uri": "ccnx:/nict.go.jp/kanribus/telemetry/direct/ns/ns-cnf-robotshop-201/type/cnf/metric/cpu_usage/latest/7200s/ave/10s/", "metric": "cpu_usage"}
処理時間：12～13秒程度
CPU: Intel Core i9 14900K
GPU: GeForce 4060
Copyright © National Institute of Information and Communications Technology. All Rights Reserved.

# 動作環境

- PC構成
  - Ubuntu 24.04LTS, Intel Core i9 14900K, GeForce 4060, 64GB Memory, ...

- LLMについて
  - Ollamaでマイクロソフト社(MS)のLLM, phi4を使用 (ローカルで実行)
    - Ollama: ローカルに言語モデルをDLして実行可能なソフトウェア
      - REST APIに対してプロンプト(LLMへの依頼文、JSON形式)をPOSTし、分析結果を取得
  - MSのphi4, phi4-mini, GoogleのGemma3n, MetaのLlama2などをテストした結果、phi4を選択
  - 命令文、プロンプトの書き方で結果が大きく揺らぐ
    - 5 mins. → 5 secs.と解釈する、”平均”の指示を誤認する、など

# プロンプトの例

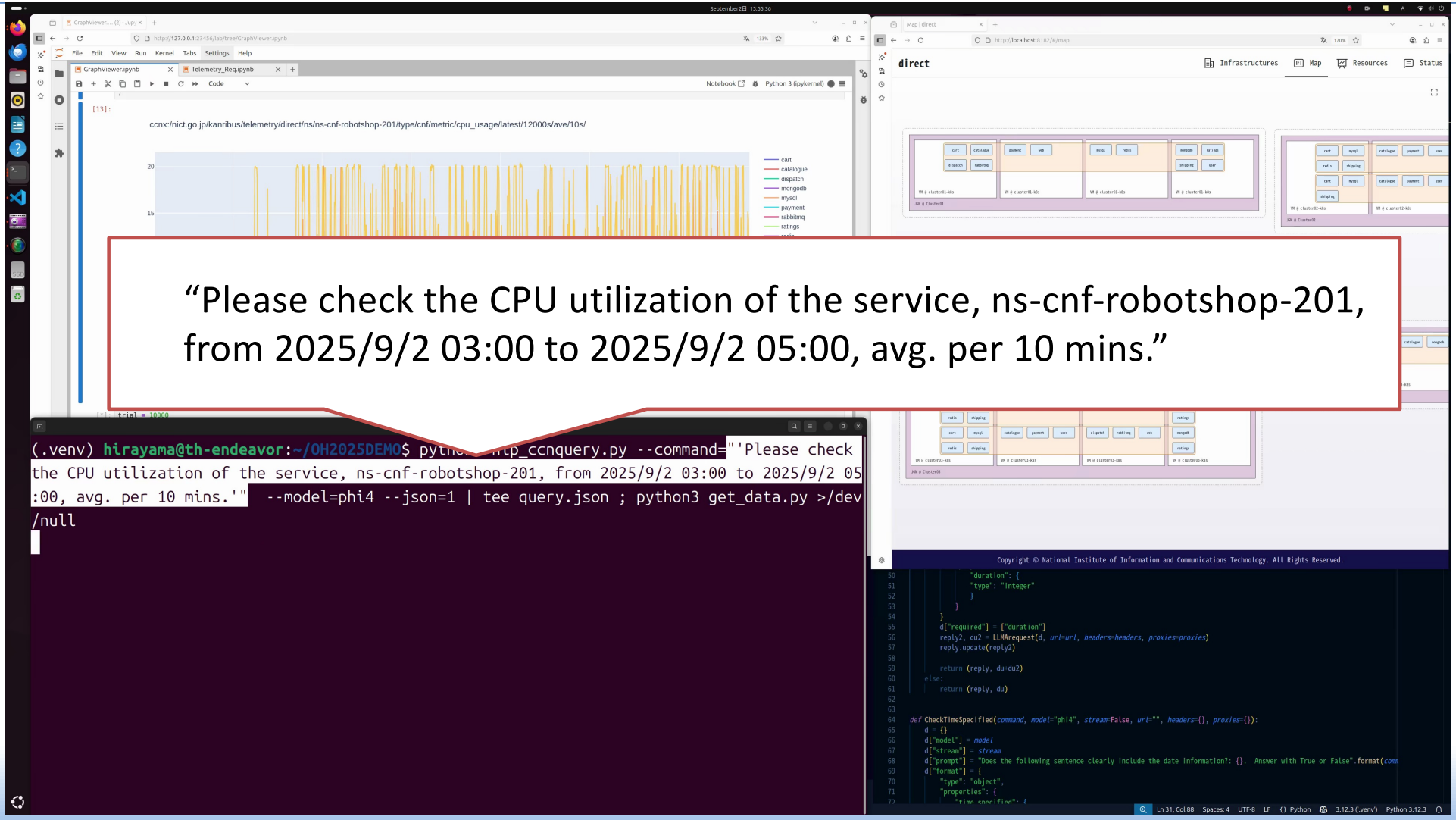
- 時刻が指定されている場合に、
  日時をYYYYMMDDHHMMSS形式で
  返すよう指示する場合

- phi4の場合、
  - “2025/9/4 18:00”
    →20250904180000 のように
    SS部分を補完してくれる
  - 表記のゆらぎに対応可能
    cf. datetime.strptime (Python)
    - 定義(Ex. ‘%Y/%m/%d %H:%M’)通りに
      記述しないと正確に読み取れない

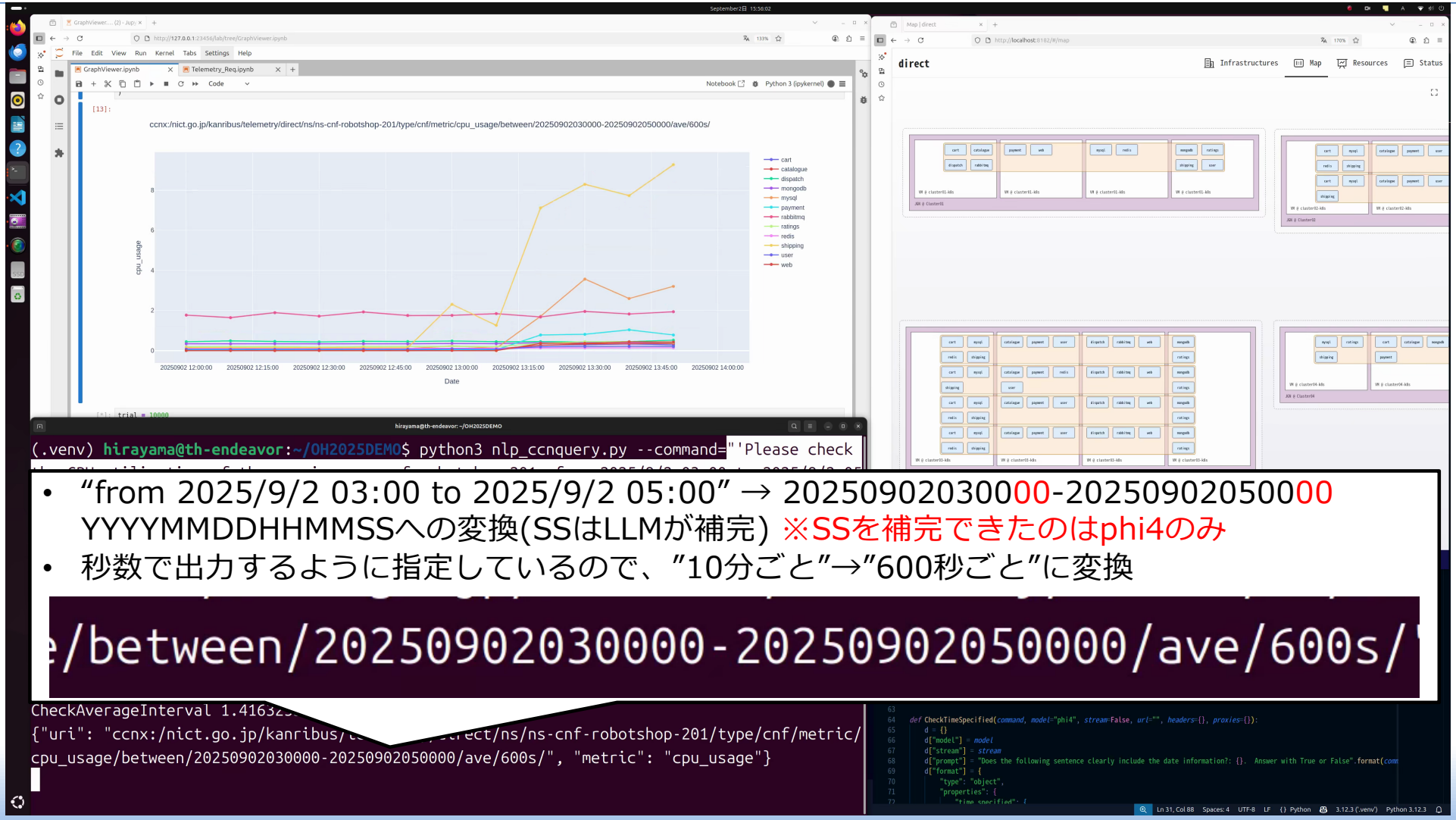
```
d[“prompt”] = “Extract the information of time period as
                ‘YYYYMMDDHHMMSS’ format from the
                following sentence {命令文}"
d[“format”] = {
   “type”: “object”,
   “properties”: {
      “start_time”: {
      “type”: “string”
      },
      “end_time”: {
      “type”: “string”
      }
   }
}
d[“required”] = [“start_time”, “end_time”]
                ↑いつから   ↑いつまで
```

抜き出してほしい情報の定義

# 時刻指定した場合の動作例(1/2)



# 時刻指定した場合の動作例(2/2)




- “from 2025/9/2 03:00 to 2025/9/2 05:00” → 20250902030000-20250902050000
  YYYYMMDDHHMMSSへの変換(SSはLLMが補完) ※SSを補完できたのはphi4のみ
- 秒数で出力するように指定しているので、”10分ごと”→”600秒ごと”に変換

e/between/20250902030000-20250902050000/ave/600s/

# まとめ

- ネットワーク基盤の管理制御の自動化
  - AIによるVNF/CNF基盤の需要予測、リソース制御の自動化
  - 障害検知→サービス退避の自動化 (共同研究)

- 基盤内の情報を正確・迅速に取得するテレメトリ技術の検討
  - IBNにおいても、状況の把握は重要
  - CCNベースの完全分散型テレメトリ情報共有基盤
    - 負荷分散、余剰な情報の削減、低遅延

- LLMによる自然言語→データ取得URIの変換
  - 煩雑な記述ルールを覚えずに使用できる
  - 不足情報(日時)の補足や、表記のゆらぎへの対応が可能