働けAI Agent！

NW障害時切り分け支援の実装と有用性評価

サービスインフラCBUインフラCBUネットワークユニットネットワーク1ディビジョン

新納和樹

岡田嘉

LY Corporation

Agenda

お話の前に

Clos NWの障害確認の難しさ

スクリプトからAI Agentへ

実装と評価

新納 和樹
Kazuki Shinno

サービスインフラCBUインフラCBUネットワークユニット
ネットワーク1ディビジョン

2020 – 2022

- コンテンツ系NW
    - BB/Service NW構築、運用、監視周り担当

2022 –

- LINEヤフー(旧ヤフー株式会社)
    - Clos NW/AWS 設計、構築、運用
    - 自動化ツール開発

岡田嘉
Yoshimi Okada

サービスインフラCBUインフラCBUネットワークユニット
ネットワーク1ディビジョン

– 2022

- NW機器ベンダー、モバイルキャリア
    - キャリアグレードなネットワークに関わる

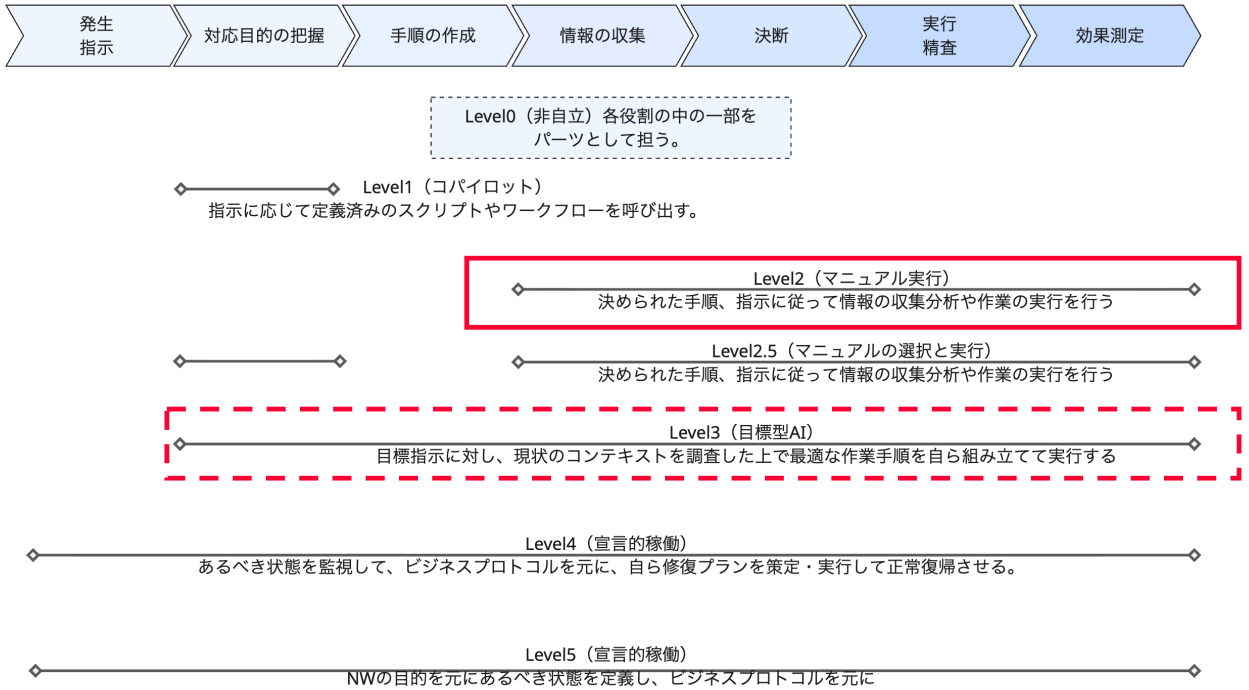
2023 –

- LINEヤフー(旧ヤフー株式会社)
    - Clos NW 設計、構築、運用
    - 自動化ツール開発
    - ネットワークエンジニア

お話の前に

AI Agentの独立性のレベル分けと今回のPJのターゲット

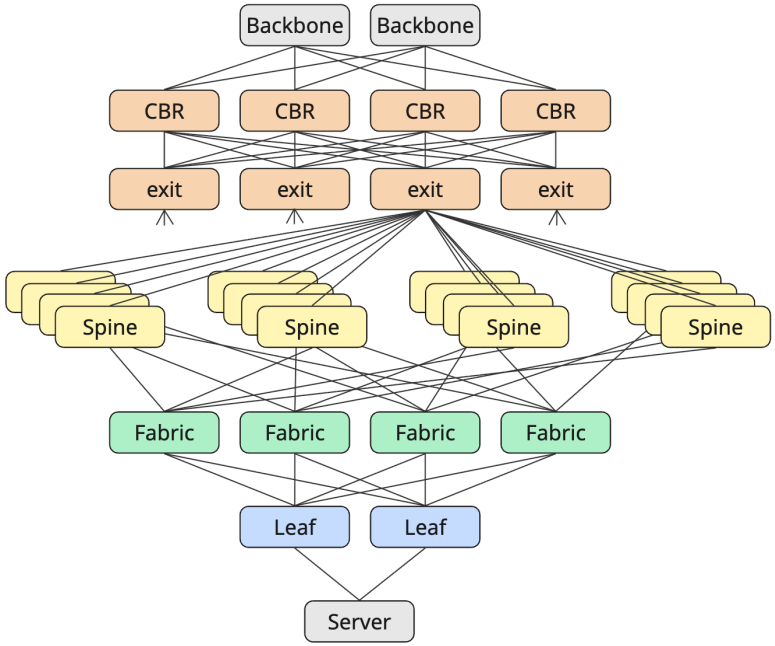
AI Agentの自立性についてちょっと考えてみました



Clos NWの障害確認の難しさ

Clos NWの障害確認の難しさ

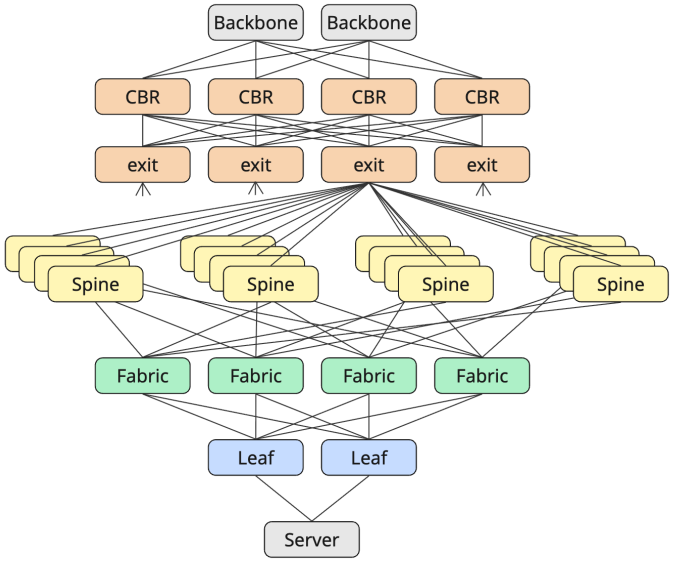
ネットワーク運用の悩み



アラートは出てないので、大丈夫（多分、、、）です！

Clos NWの障害確認の難しさ

弊社のNetworkをおさらい



堅牢

- Clos Network
- x2-x8 冗長で強固（Spine層はマルチプレーン冗長も）
- シンプルなIP Forwardingに専念
- ハッシュは均等性を重視した5tupleを利用

コスト重視

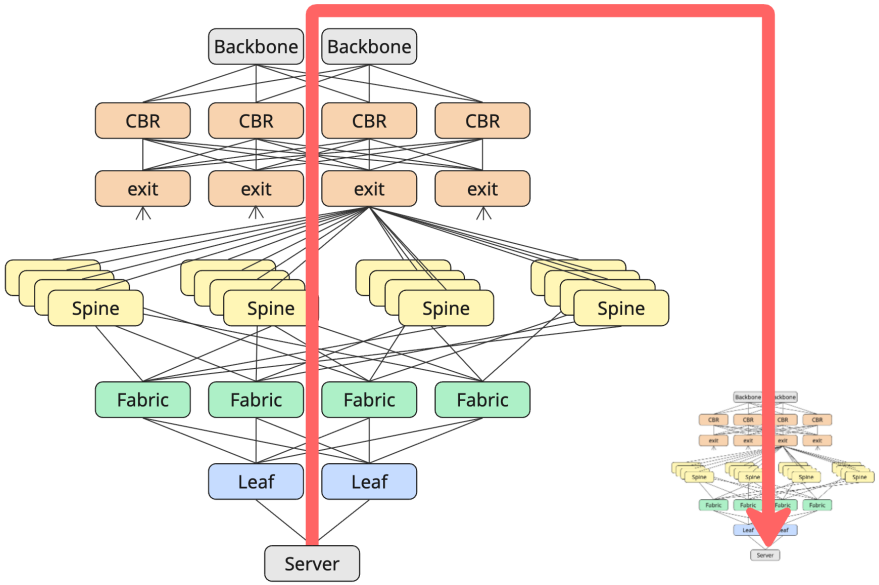
- ホワイトボックスを含む、複数ベンダーのスイッチ、光モジュールを採用

クラウド基盤

- 常にVMの改廃/移動/再起動が行われる

Clos NWの障害確認の難しさ

登場人物多すぎ問題



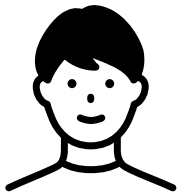
- 関連するスイッチが多い
  - Server-Server間の疎通を確認する際に転送に関わっているノードがとても多い
  - リージョンを跨ぐと転送に関わっているClosのスイッチだけで最大60ノードに及ぶ
  - パスが多くて(最大数万通り)Pingでは確認が不可能
- クラウドのしんどさ
  - サービス単位で申告が来た際にはさらにDC内にインスタンスが点在する

障害が発生した時に職人の勘による切り分けが必要
漠然とした問い合わせに自信を持って問題なしと言えない。。

スクリプトからAI Agentへ

Clos NWの障害確認の難しさ

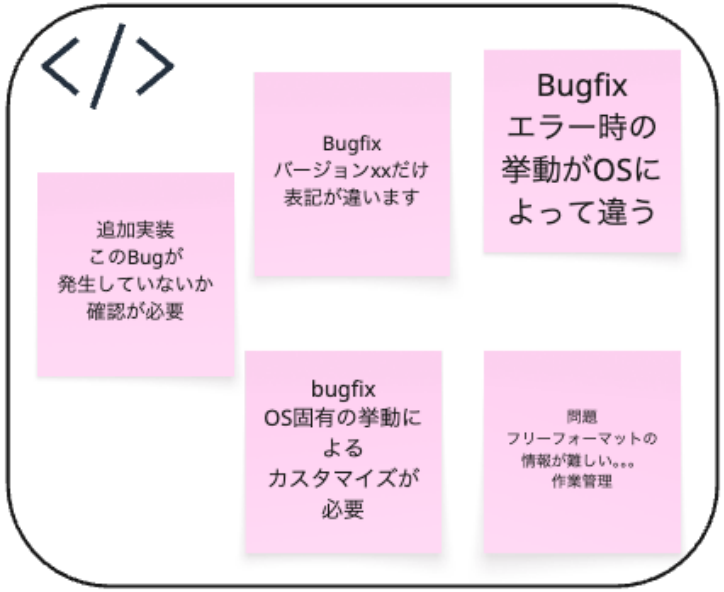
よしスクリプトにやらせよう！



全てのノードを確認する人手があればいいのになあ

Clos NWの障害確認の難しさ

よしスクリプトにやらせよう！

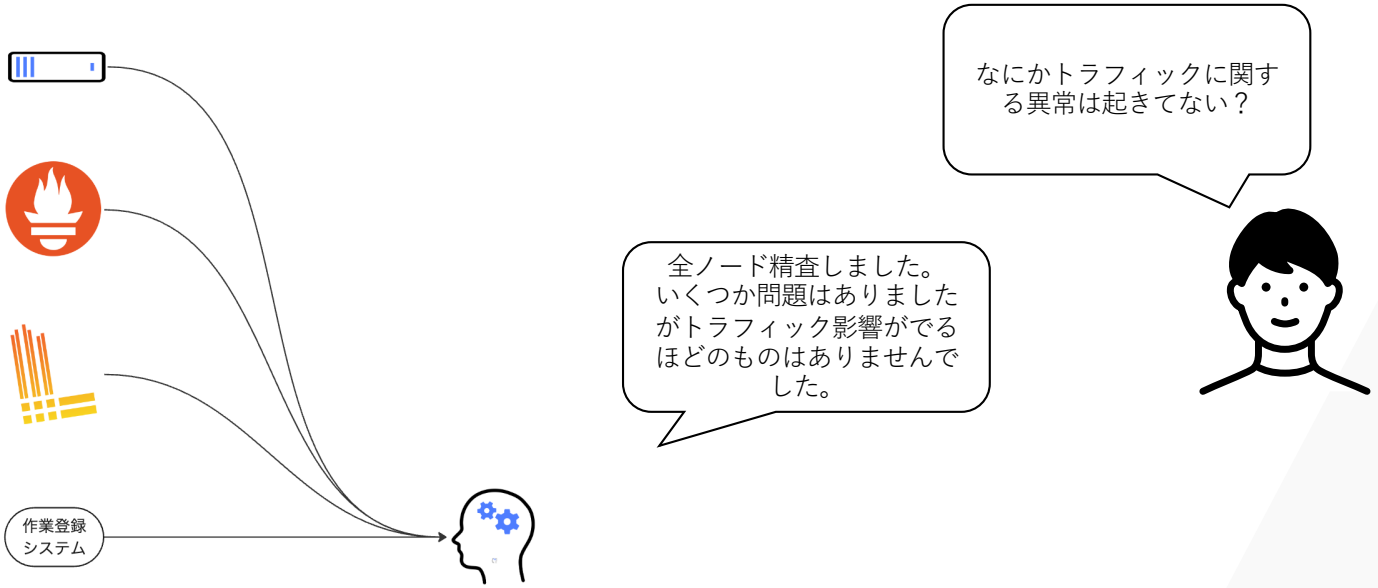


- スクリプトの管理修正は結構大変
  - 想定外のパターンが出るたびに修正地獄
  - バージョンアップで出力が変わる
  - 障害確認系のコマンドは抽象化されてない。
- もう少し柔軟に判断してほしい
  - 暫定構成やメンテナンスなど全部異常として判定
  - その結果を精査するのも大変

もう少し、柔軟な判断をしてくれないかなあ。。。

よし、AIにまかせよう

AIに調査させてAIに精査させる仕組み

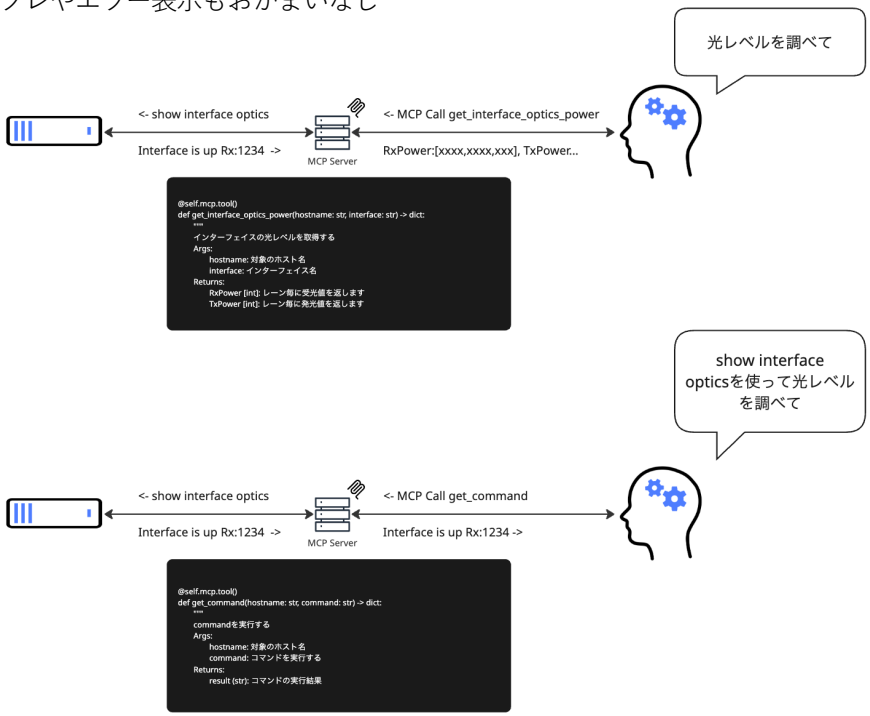


実装概要

AIにできることはAIに

コードはベースに徹して汎用性をあげてAIに任せる！

表記のブレやエラー表示もおかまいなし



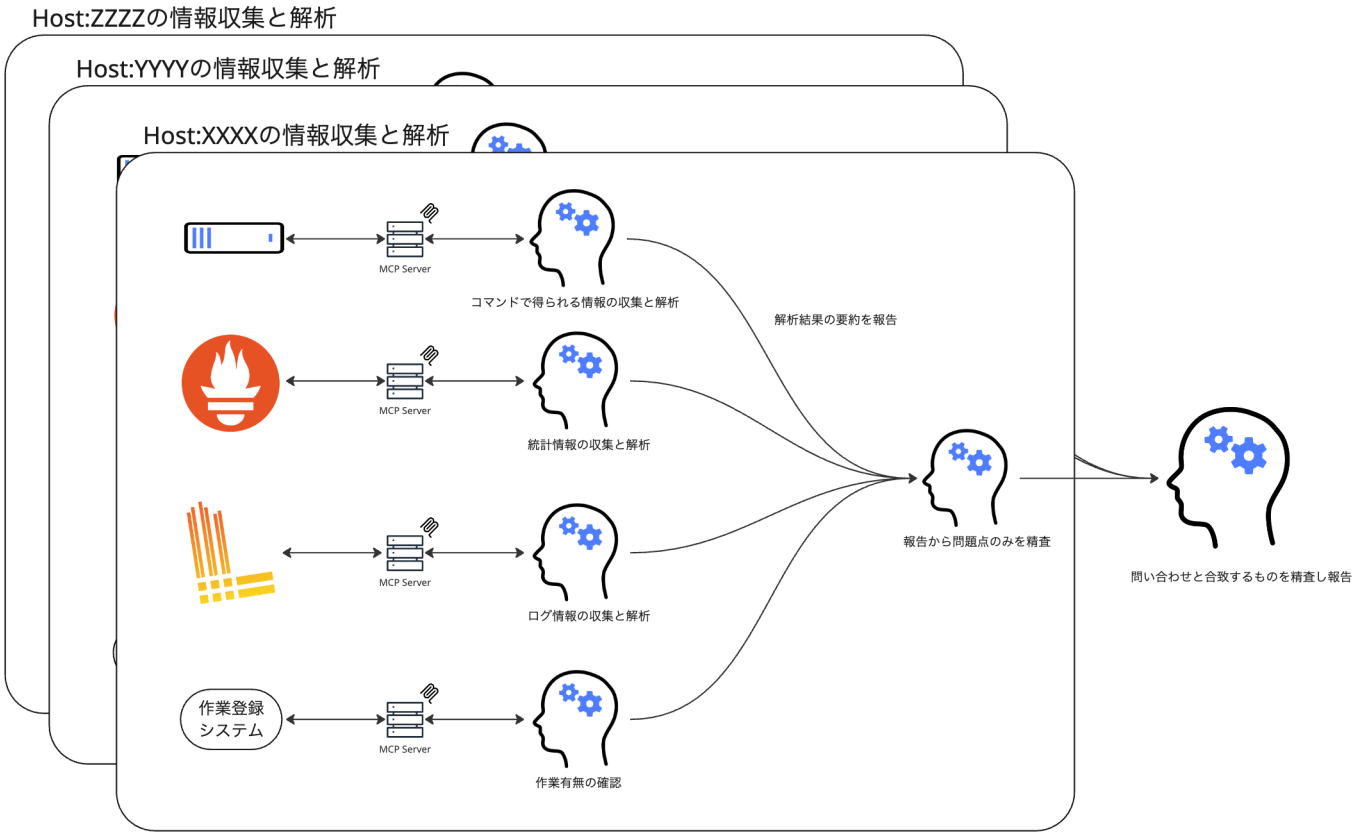
確認事項などを自然言語で表記

```
"matcher": r"A社装置.*",
"procedure": (
  "【A社BGP接続性診断】"
  "0. **重要**: lokiツールで指定時間帯のログを確認し、BGP neighbor down/up、
      経路変動の発生を把握。prometheusツールでBGPネイバー状態・経路数の推
      移も確認"
  "1. 'show ip bgp summary' でBGPネイバー状態を確認"
  "2. 全てのUPLINKネイバーがEstablished状態であることを確認"
  "3. 'show ip route 0.0.0.0' でdefault route (0.0.0.0/0) が存在することを確認"
  "4. default routeのnext-hopがUPLINKルーターであることを確認"
  "5. 'show ip bgp 0.0.0.0' でdefault routeをDOWNLINKへ広告しているか詳細
      確認（Advertised toにDOWNLINKネイバーが含まれているか）"
  "6. BGPセッションが確立していない、またはdefault routeが広告されていない場
      合は、該当ネイバーのログを確認して原因を特定"
)
```

実装と評価

実装と評価

マルチエージェント構造を導入



実装と評価

なぜマルチエージェントなのか

コンテキストが爆発する

- コンテキストが拡大すると分析能力が落ちたり、暴走したり、、、

指示していないコマンドを打ち出すエージェントくん

```
WARNING - POLICY VIOLATION (Not Whitelisted): host=hostA, cmd='nv show router bgp'
WARNING - POLICY VIOLATION (Not Whitelisted): host=hostB, cmd='net show router bgp summary'
WARNING - POLICY VIOLATION (Not Whitelisted): host=hostC, cmd='net show router bgp summary'
```

実行速度

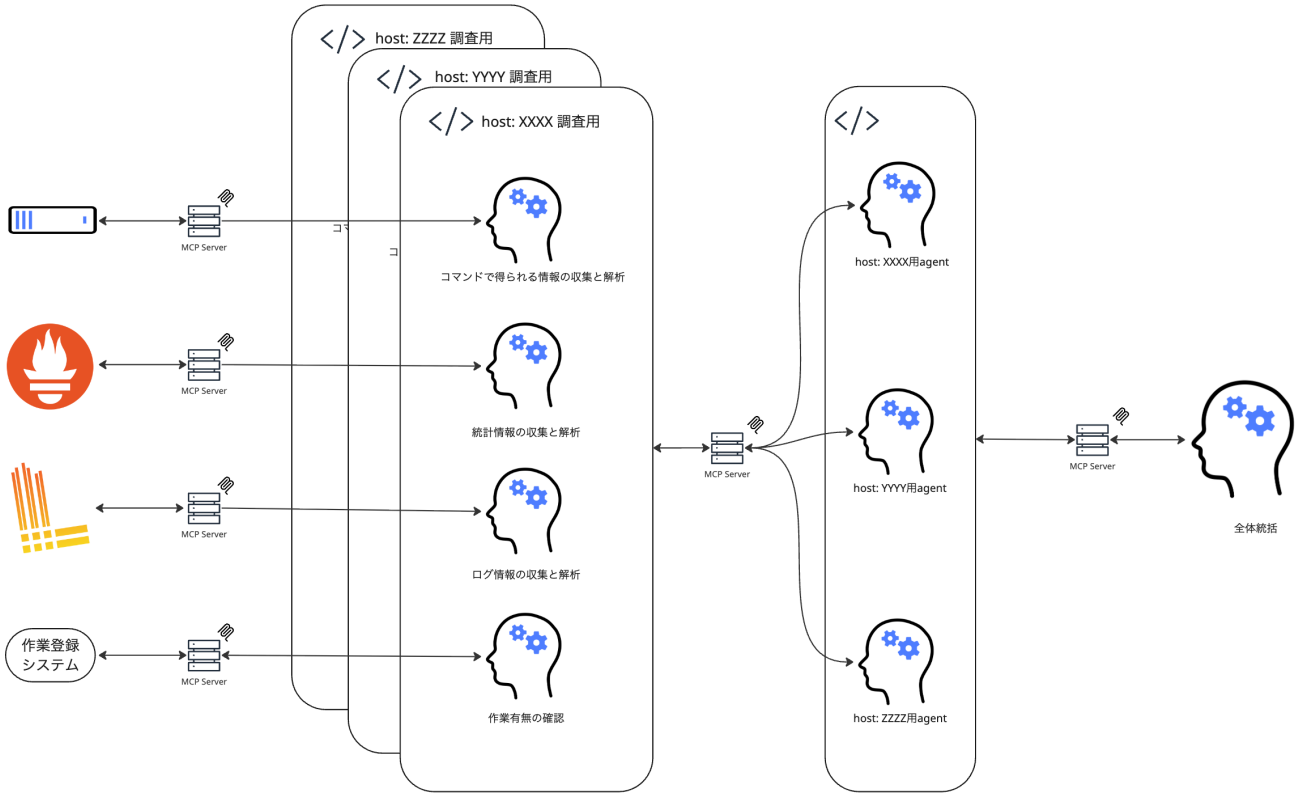
- AIは基本的にシーケンシャルにしか動いてくれない

	1ノード	5ノード	20ノード（理論値）
単一Agent	35.2秒	76.4秒	227.1秒
マルチエージェント	57.2秒	85.9秒	193.5秒

5回実行の平均で計算

実装と評価

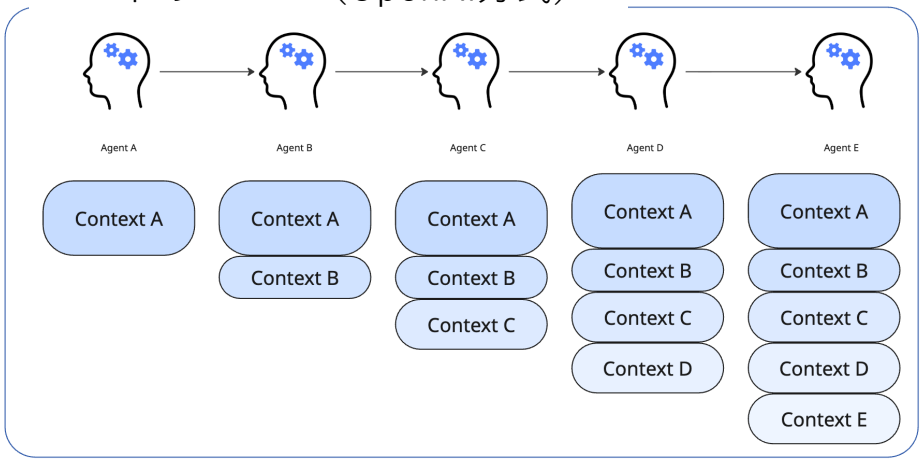
詳細なアーキテクチャ



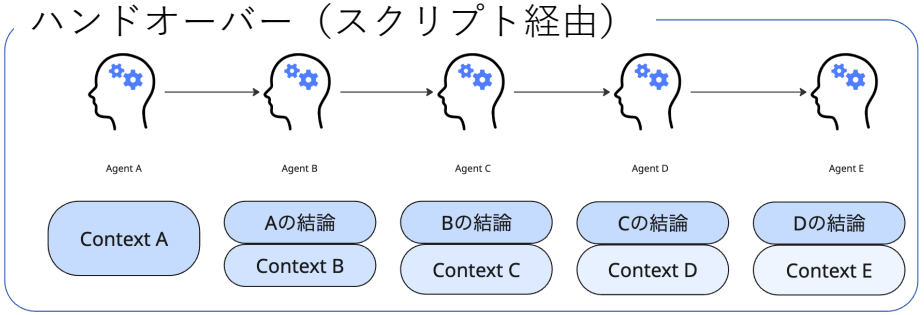
実装と評価

ハンドオーバーとの比較

ハンドオーバー（OpenAI方式）



ハンドオーバー（スクリプト経由）

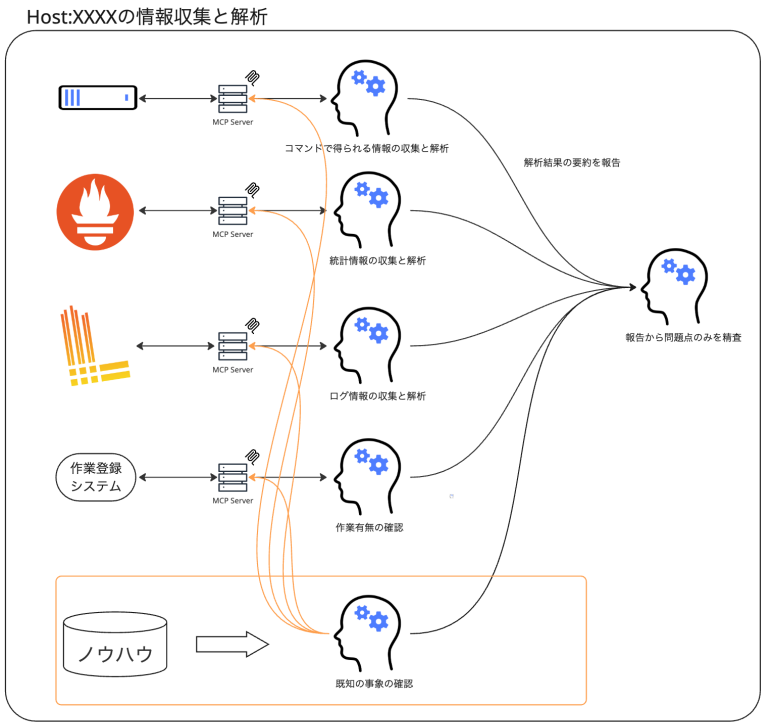


OpenAIのハンドオーバーはコンテクストを基本的に受け継ぐのでコンテクストが拡大する。
コンパクションが入るが、その過程で大切な情報が抜け漏れすることも、、

スクリプトによる呼び出しを挟むことで、都度、任意の要約を実施できる

実装と評価

Level3への挑戦と限界

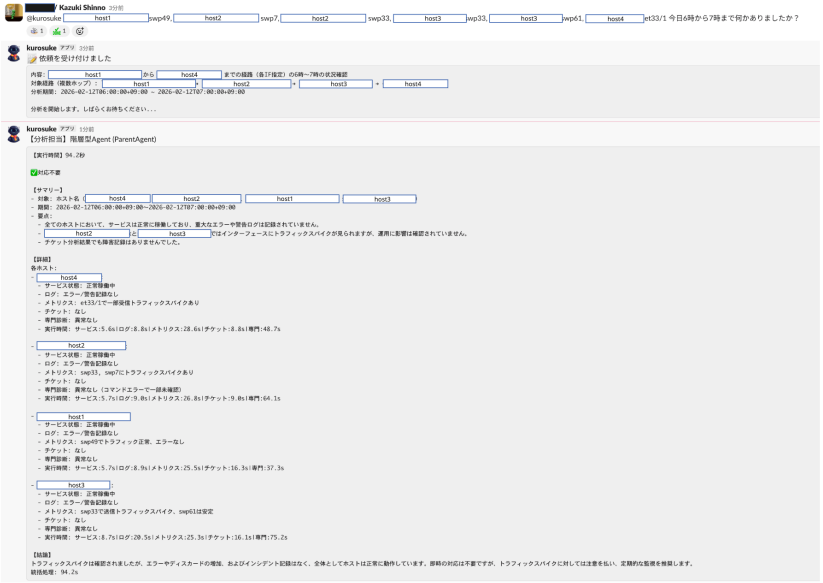


自走の限界

多少のプロンプトでの介入は行いつつも、精査はAIに任せていた。
→特定のバージョンのバグ、起きがちなイシューについての検知は難しかった。

実装と評価

有用性について


Kazuki Shinno 3分前
@kurosuke host1 swp49, host2 swp7, host2 swp33, host3 swp33, host3 swp61, host4 et33/1 今日6時から7時まで何かありましたか？
1 1
kurosuke アプリ 3分前
依頼を受け付けました
内容: host1 から host4 までの経路（各ISP指定）の6時～7時の状況確認
対象経路（複数ホップ）: host1 → host2 → host3 → host4
分析期間: 2026-02-12T06:00:00+09:00 ～ 2026-02-12T07:00:00+09:00
分析を開始します。しばらくお待ちください...
kurosuke アプリ 1分前
【分析担当】階層型Agent (ParentAgent)
【実行時間】94.2秒
対応不要
【サマリー】
- 対象: ホスト名 (host4, host2, host1, host3)
- 期間: 2026-02-12T06:00:00+09:00～2026-02-12T07:00:00+09:00
- 要点:
  - 全てのホストにおいて、サービスは正常に稼働しており、重大なエラーや警告ログは記録されていません。
  - host2 と host3 ではインターフェースにトラフィックスパイクが見られますが、運用に影響は確認されていません。
  - チケット分析結果でも障害記録はありませんでした。
【詳細】
各ホスト:
- host4
  - サービス状態: 正常稼働中
  - ログ: エラー/警告記録なし
  - メトリクス: et33/1で一部受信トラフィックスパイクあり
  - チケット: なし
  - 専門診断: 異常なし
  - 実行時間: サービス:5.6s | ログ:8.8s | メトリクス:28.6s | チケット:8.8s | 専門:48.7s
- host2
  - サービス状態: 正常稼働中
  - ログ: エラー/警告記録なし
  - メトリクス: swp33, swp7にトラフィックスパイクあり
  - チケット: なし
  - 専門診断: 異常なし（コマンドエラーで一部未確認）
  - 実行時間: サービス:5.7s | ログ:9.0s | メトリクス:26.8s | チケット:9.0s | 専門:64.1s
- host1
  - サービス状態: 正常稼働中
  - ログ: エラー/警告記録なし
  - メトリクス: swp49でトラフィック正常、エラーなし
  - チケット: なし
  - 専門診断: 異常なし
  - 実行時間: サービス:5.7s | ログ:8.9s | メトリクス:25.5s | チケット:16.3s | 専門:37.3s
- host3:
  - サービス状態: 正常稼働中
  - ログ: エラー/警告記録なし
  - メトリクス: swp33で通信トラフィックスパイク、swp61は安定
  - チケット: なし
  - 専門診断: 異常なし
  - 実行時間: サービス:8.7s | ログ:20.5s | メトリクス:25.3s | チケット:16.1s | 専門:75.2s
【結論】
トラフィックスパイクは確認されましたが、エラーやディスカードの増加、およびインシデント記録はなく、全体としてホストは正常に動作しています。原時の対応は不要ですが、トラフィックスパイクに対しては注意を払い、定期的な監視を推奨します。
統括処理: 94.2s

ごめんなさい

稼働開始から数ヶ月、、、
実際に動かすようなシチュエーションが起きませんでした。。

動きます

あえて、問題を起こすとちゃんと確認と事象の検知はできます

実装と評価

LLM as interpriterとして

実際AIにできることをAIにやらせてみてどうだったか

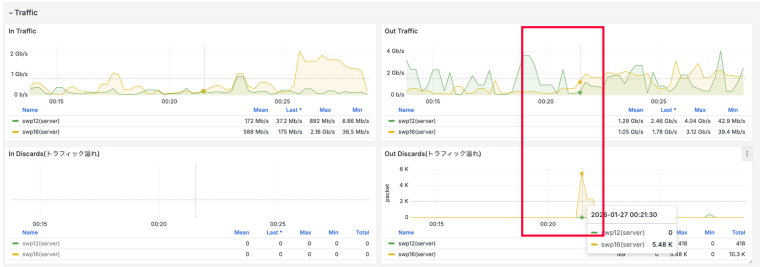
- コマンドのパース精度は問題ない
- モデルによって複雑な指示の追従性に問題が生じた
- プロンプトの設計はAIに任せると質が高い
- コード部分のメンテナンスは不要になった
- 周辺インフラ（MCPのスクリプト開発コストやインフラ周り）は以前として必要

実装と評価

見えてきた問題点

AIの決断のためには情報の粒度が荒い

- 急激なトラフィック溢れについて、分単位のポーリングでは可視化できない事がある。
- 人間はこの時、収容サーバーなどを考慮して決断的に判断できるが、AIはこれができない。



マイナーツールの維持にかかる負担

- メンテナンスはノウハウの自然言語での更新だけだが、それを持っても単機能のツールを維持する工数を捻出する必要はある。
- 実際に仮運用を開始して、1-2月で運用機会が数回程度
- 他のツールとノウハウ部分の共有や、ほぼメンテナンスフリーとなるLv3-4への進化を目指さなければならない

議論したいこと

- AI Agentによる障害検知に実用化の未来はあると思いますか？
- LLM-as-Interpreter vs スクリプト、スイッチの確認作業はどちらであるべきだと思いますか？
- ”知恵注入方”と”自立思考型” どちらがより実用性があるか各社の事情都合などと絡めてご意見交わしたいです