Astemo
Mobility Beyond

Software-Defined Vehicle 向けの クラウドネイティブ開発環境における Determinism の改善

Astemo株式会社
技術開発統括本部
シニアダイレクター
石郷岡 祐

ROSConJP 2025

Astemo
Mobility Beyond

Contents

1. 自己紹介、会社紹介
2. Software-Defined Vehicleとクラウドネイティブ開発とは
3. Determinism(決定論)と課題
4. 解決のアプローチとROS2への適用方針
5. 適用評価
6. まとめ

自己紹介

石郷岡　祐

いしごうおか　たすく

- 博士(情報学)
- 書籍：機能安全の基礎と応用
- SOAFEE/COVESA/AWF/
  Open SDV Initiativesメンバー
- ROSConJP2025プログラム委員

2008 **株式会社日立製作所　研究開発グループ　入社**

自動車：AUTOSAR Classic Platform、機能安全、マルチコアソフトウェア

2013 **Hitachi Europe GmbH （ドイツ：ミュンヘン）**
Transportation, Energy, and Environment Laboratory　出向

自動車： 機能安全

2015 **株式会社日立製作所　研究開発グループ　帰任**

自動車：AUTOSAR Adaptive Platform、Autoware Foundation
物流：Robot Operating System
インフラ：エッジクラウドコンピューティング

2022 **日立Astemo株式会社　技術開発統括本部　出向**

自動車：Software-Defined Vehicle、オープンイノベーション

2025 **Astemo株式会社　技術開発統括本部　転籍**

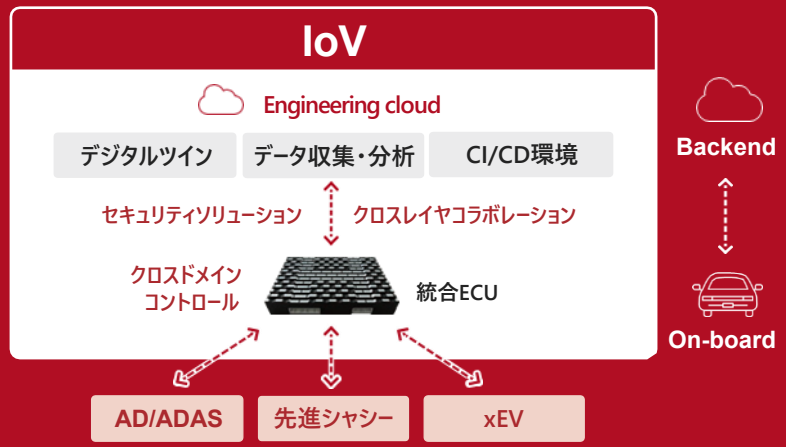
自動車：Software-Defined Vehicle、オープンイノベーション、AIエージェント

Astemoの事業分野

Astemo
Mobility Beyond

Solutions + Products

クラウド開発環境と高性能な
コンポーネントを活かした
クロスドメイン制御でSDV時代に
最適なソリューションを提供します。



Solutions

Products


自動運転・
先進運転支援システム


パワートレインシステム


シャシーシステム


二輪車用システム


汎用製品・産業機器


アフターマーケット・
整備用品

詳細は
Webサイトへ

Astemo
Mobility Beyond

Contents

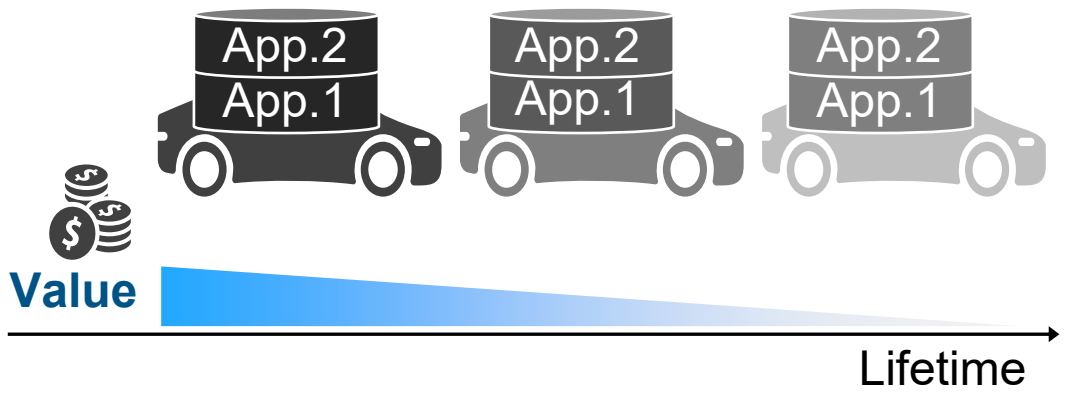
1. 自己紹介、会社紹介
2. Software-Defined Vehicleとクラウドネイティブ開発とは
3. Determinism(決定論)と課題
4. 解決のアプローチとROS2への適用方針
5. 適用評価
6. まとめ

Software Defined Vehicle(ソフトウェア定義車両、SDV)とは

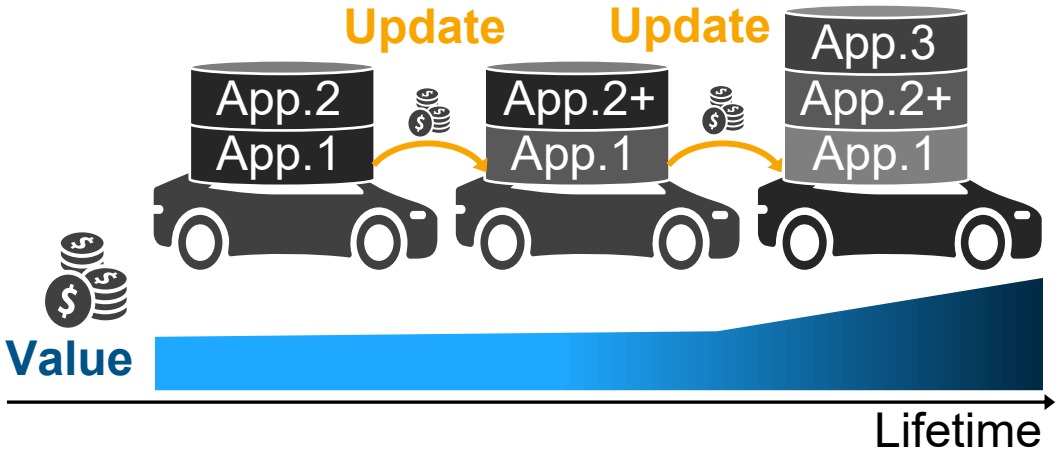
Astemo
Mobility Beyond

■ SDV：クラウドとの通信により、自動車の機能を継続的にアップデートすることで、運転機能の高度化など従来車にない新たな価値が実現可能な次世代の自動車[1]。効率的なアプリケーション開発が重要となる。

従来のクルマ



Software-Defined Vehicle



[1]「モビリティDX戦略」2025年のアップデート、経済産業省、
https://www.meti.go.jp/policy/mono_info_service/mono/automobile/jido_soko/pdf/mobilitydxsenryaku2025.pdf

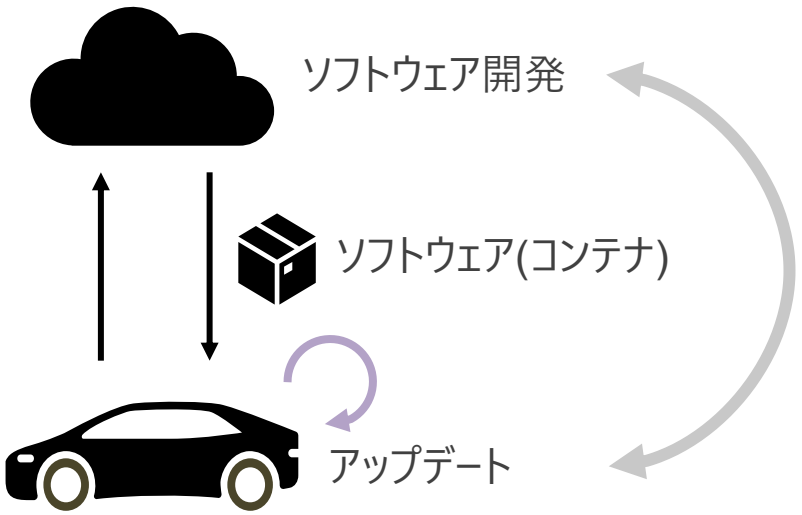
SDVコミュニティ：SOAFEE

SOAFEE: Scalable Open Architecture For Embedded Edge
EWAOL: Edge Workload Abstraction and Orchestration Layer

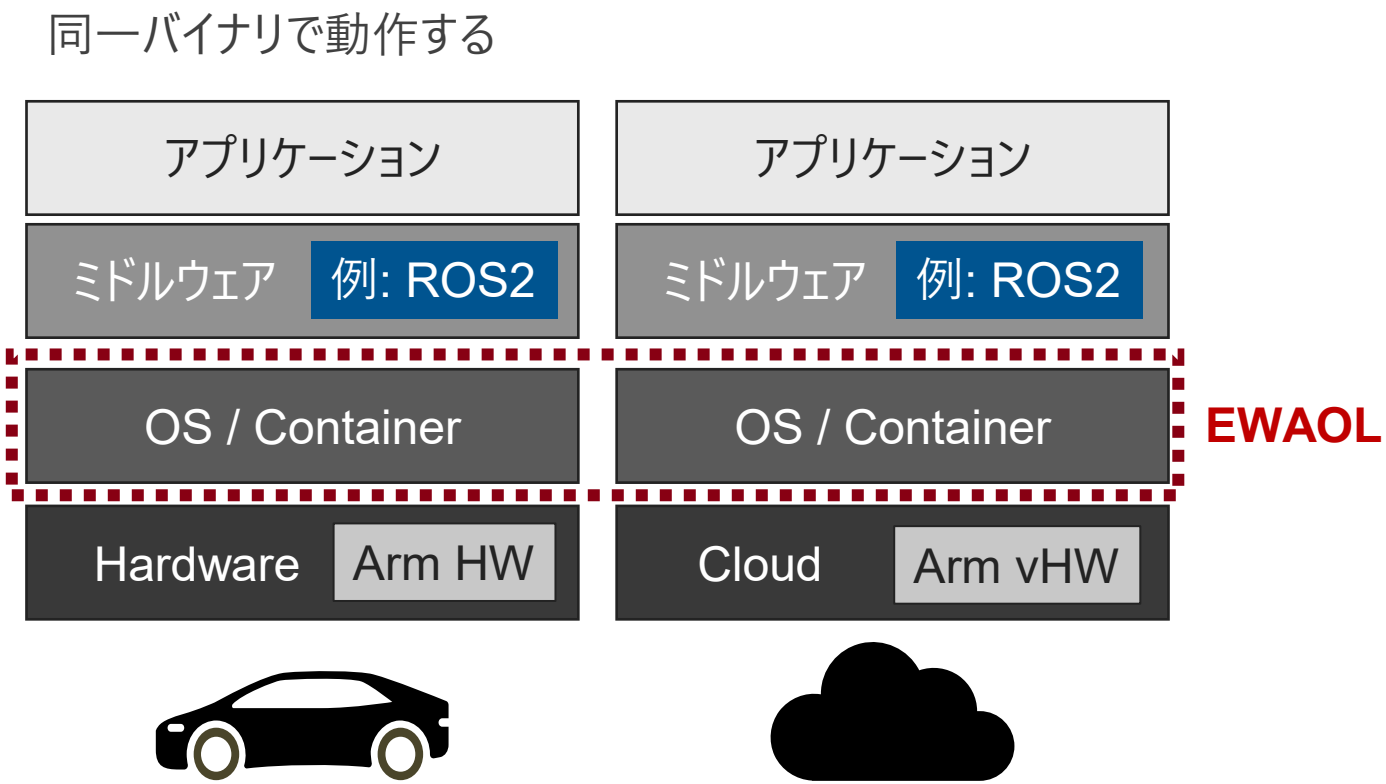
Astemo
Mobility Beyond

■ 産業界主導のSDV実現のためのOSSプロジェクト。Arm他が中心メンバとなり、世界中100社以上が参画。
■ 車両とクラウドの実行環境の同一(cloud-native)とし、アプリケーションの開発効率化をねらう。

SOAFEEの狙い



ソフトウェアアーキテクチャ

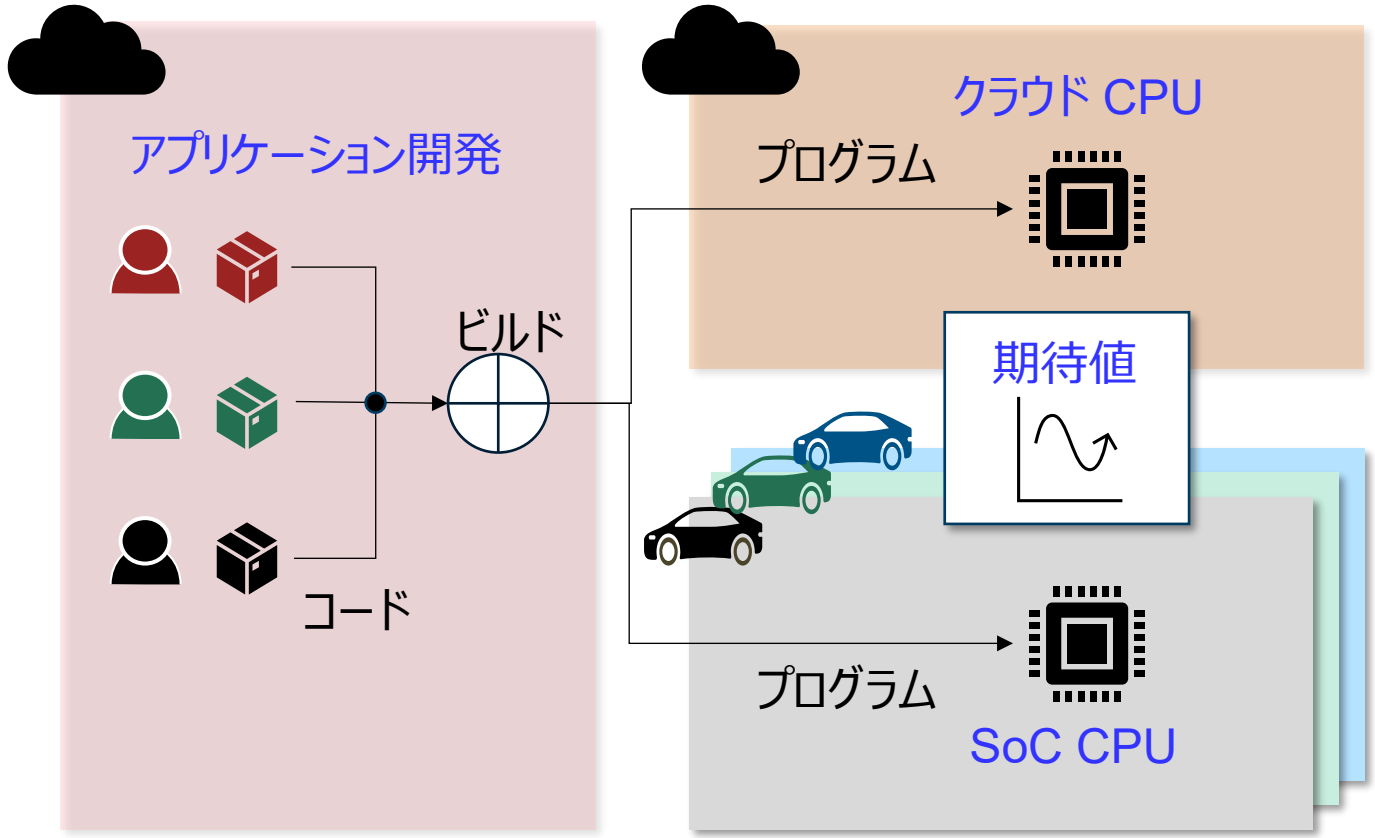


クラウドネイティブ開発における課題

Astemo
Mobility Beyond

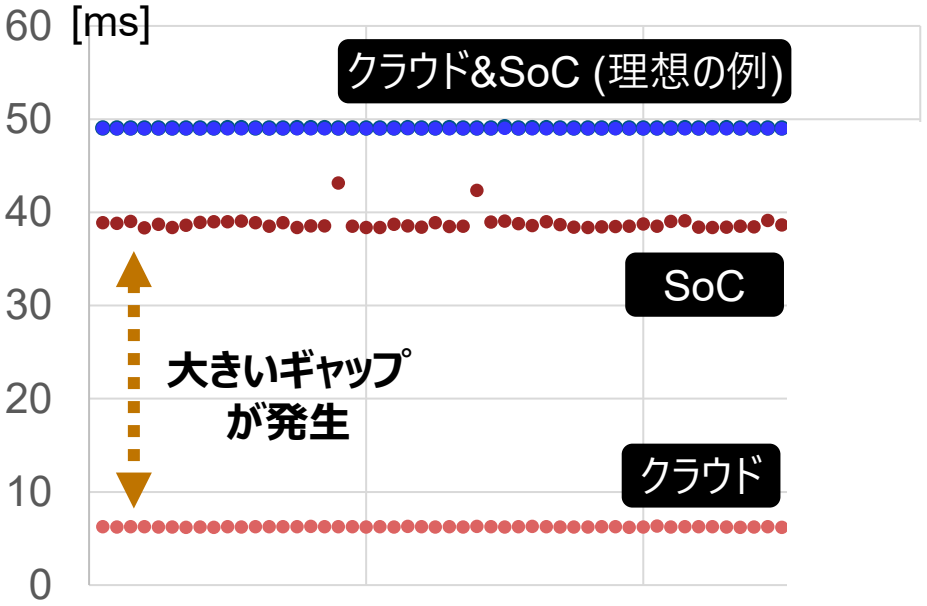
- クラウドネイティブ開発ではクラウドで様々な開発者がアプリを開発⇒テスト。テスト後にアプリがクルマに配信。
- クラウドとクルマ(SoC)ではCPU性能が異なるため、実行タイミングが異なる問題が発生。これを改善したい。

目指している姿



課題

アプリケーション実行時の出力タイミング



デッドライン以内に終わる制約を満たしつつ、
一致させたい（**determinismの改善**）

Astemo
Mobility Beyond

Contents

1. 自己紹介、会社紹介
2. Software-Defined Vehicleとクラウドネイティブ開発とは
3. Determinism(決定論)と課題
4. 解決のアプローチとROS2への適用方針
5. 適用評価
6. まとめ

Determinism(決定論)とは

[2] Edward, A. Lee, Plato and the Nerd - The Creative Partnership of Humans and Technology, Books Gateway, MIT Press, 2017.

Astemo Mobility Beyond

■ **本発表では有名な著書[2]の定義におけるDeterminismを基づいて発表させて頂く。**
■ **Determinismは予測可能性や再現性とも呼ばれる非機能要件であり、ジッタなどの指標で評価される**

Determinism(決定論)の哲学的な解釈

- 決定論という考えは諸説あり、ラプラスの悪魔(Laplace's Demon)が有名。

> もしある知性が宇宙のすべての粒子の位置と運動量を完全に知っていて、
> 物理法則をすべて理解していれば、過去・現在・未来のすべてを予測できるという考え方

- 反論例：カオス理論や量子力学の登場により、決定論的なモデル化が難しい。(例：バタフライ効果)

Determinism(決定論)と工学的な解釈

- まだ研究中の分野である。離散的な振る舞いと連続的な振る舞いの両方を取り入れた決定論的なモデルの体系は不完全。しかしながら、非決定論的なモデルは、明示的かつ慎重に使用することで、工学において役立つ。
- 予測通りに実行され、何度も再現できる性質(**予測可能性、再現性**)

■本発表の立ち位置

現実世界は、すべてがシミュレーション通りに動かないことを我々は知っているが、それでもシミュレーションは有用である。

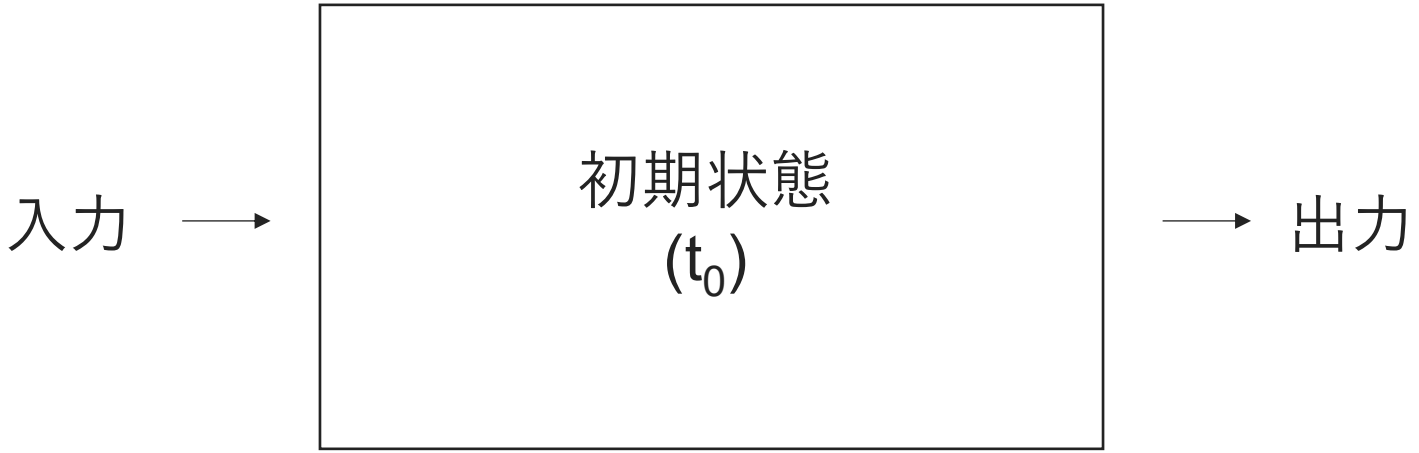
Determinismが成立する条件

[3] Time and Determinism, H. Kopetz, 2007

Astemo
Mobility Beyond

■ **Determinismが成立する条件として、[3]で記載の定義を利用する。**

時間t_0の初期状態が与えられ、入力に対して現在の状態に基づいて、次の状態と出力を生成する



自動車に当てはめたときの例

Astemo
Mobility Beyond

- 自動車は多入力・多出力であり、状態を決定するために複数のアプリケーションが連携して動作するシステム
- 「①センサの同期」、「②実行順番」、「③データの鮮度」、「④入力から出力までのレイテンシの一定化」、「⑤入力値によって出力値が１つに決まるアルゴリズム」がDeterminismを実現するために重要

入力
(Sensor)

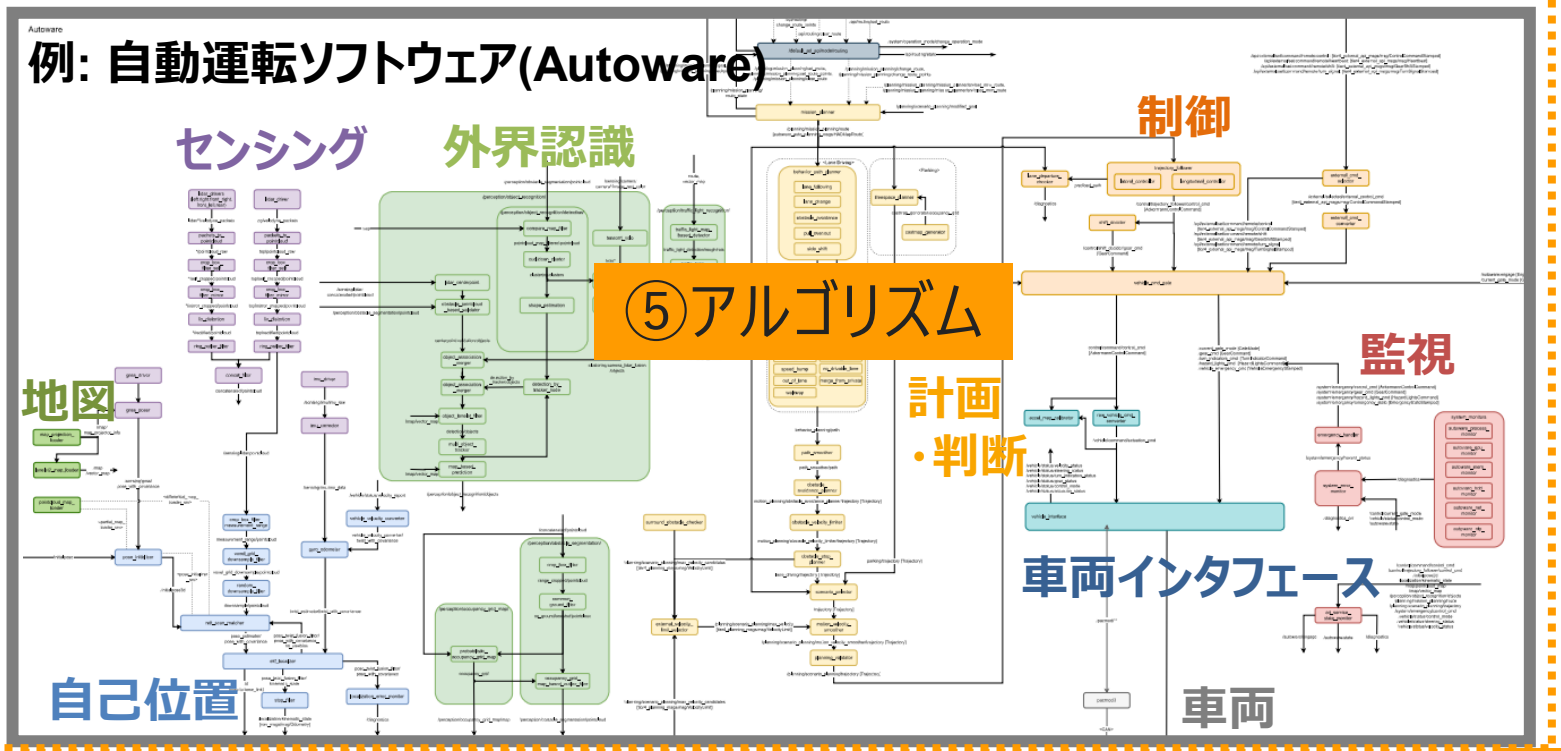
- カメラ
- Lidar
- GPS/IMU
- 車両状態
(速度等)

①センサの同期

②実行順番

初期状態(t_0)

④レイテンシの一定化



出力
(Actuator)

- モータ/
エンジン
- ブレーキ
- ステアリング

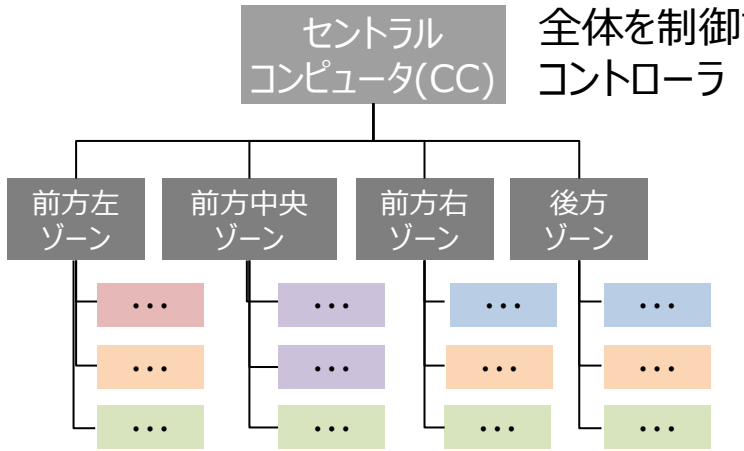
③データの鮮度

Determinism観点での改善課題

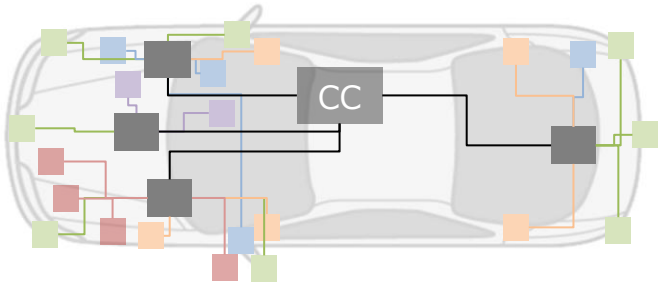
Astemo
Mobility Beyond

■ 前述Determinism要件を守る必要があるが、自動車は複数のコントローラがNWを介して通信する分散システム。
■ 近年はSoCの**高機能化**により、**実行時間の最短と最長の差が大**。SoCとクラウドHWだと差が顕著(**改善課題**)。

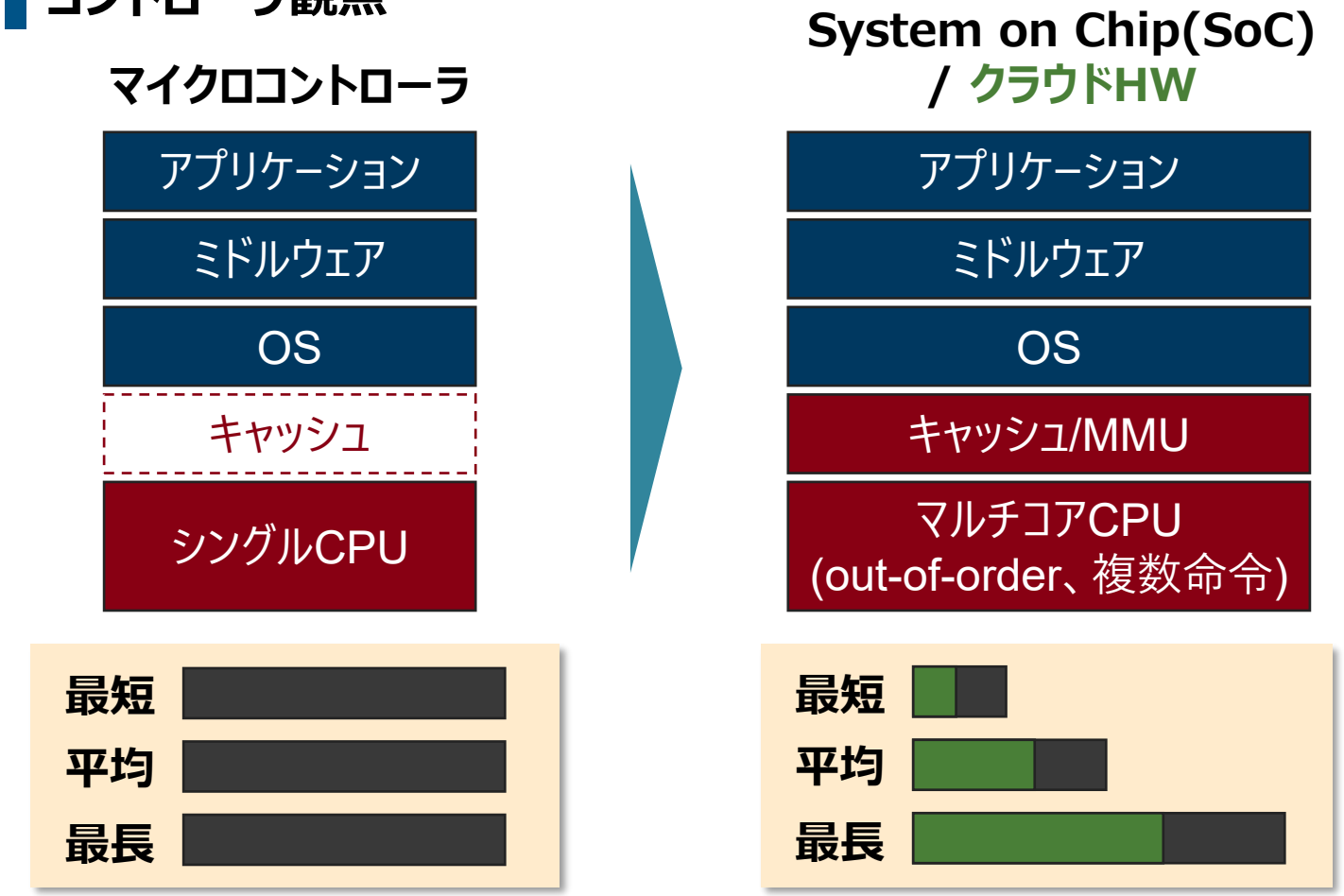
システム観点



センサ・アクチュエータを制御するコントローラ群



コントローラ観点



AD: Autonomous Driving, ADAS: Advanced Driver-Assistance Systems

本発表(クラウドネイティブ開発)で着目するDeterminismの課題

Astemo
Mobility Beyond

■ クラウドネイティブ開発における最小と最大のギャップが大きい事象は「同期、順序、鮮度、レイテンシ」に影響あり。
■ これらのタイミング問題を解決する案を提案する。

#	分類	要求
①	入力データ間の同期	モジュールへの入力データの組み合わせにおいて、入力タイミングを合せる
②	実行順番	指定の処理、通信を指定のタイミングで実行する
③	入力データの鮮度	計算で用いるデータ鮮度を一定にする
④	入力から出力までの レイテンシの一定化	処理時間を一定にする (アプリケーション、ミドルウェア、OS、データ通信)
		デッドライン以内にすべての処理を完了できる
		ECU内通信においてデータが壊れない、改ざんされない
		ECU間通信においてデータがロストしない、改ざんされない
⑤	入力値によって出力値が １つに決まるアルゴリズム	入力値によって出力値が1つの値に決まること

Astemo
Mobility Beyond

Contents

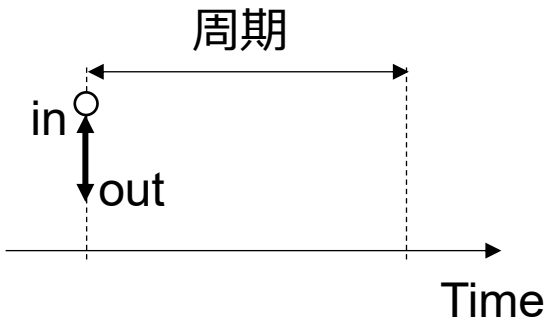
1. 自己紹介、会社紹介
2. Software-Defined Vehicleとクラウドネイティブ開発とは
3. Determinism(決定論)と課題
4. 解決のアプローチとROS2への適用方針
5. 適用評価
6. まとめ

リアルタイムプログラミングモデルの分類

Astemo
Mobility Beyond

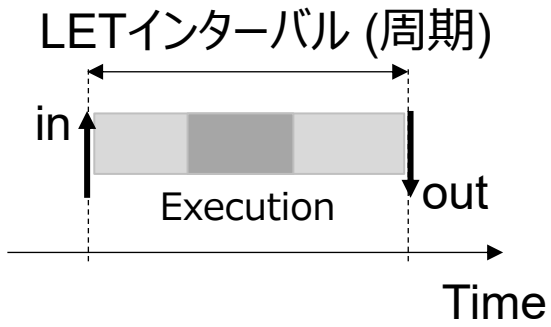
■ **タイミング起因の影響を局所化するための設計方法論として、LETに注目。**

Zero Execution Time (ZET Model)



シミュレーション等で利用
(例：Simulink)

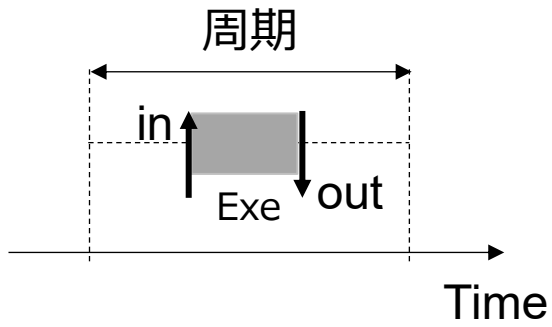
Logical Execution Time (LET Model)



計算処理(Execution)が
入力と出力処理の間で実行されればよく、
計算処理の時間変動に強い

Determinismの改善が期待できる

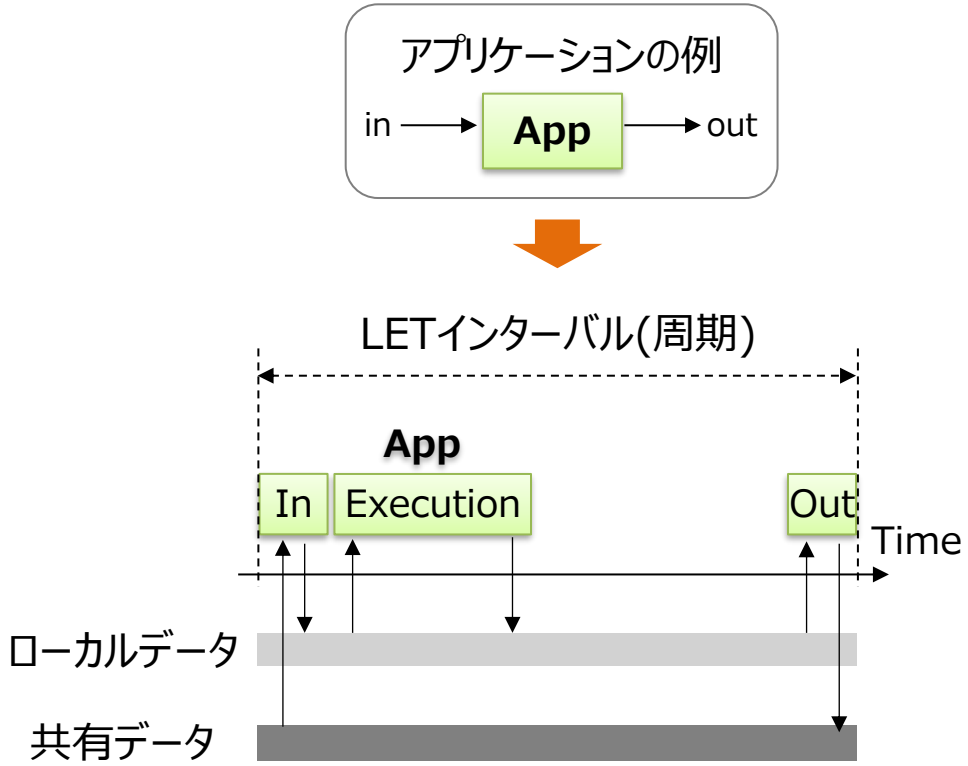
Bounded Execution Time (BET Model)



影響が伝搬

従来手法： LETのデータ通信

■ LETではアプリケーションを**入力処理、計算処理、出力処理に分離**。
■ 周期の最初に共有データを参照(Copy-in)し、最後に更新(Copy-out)する



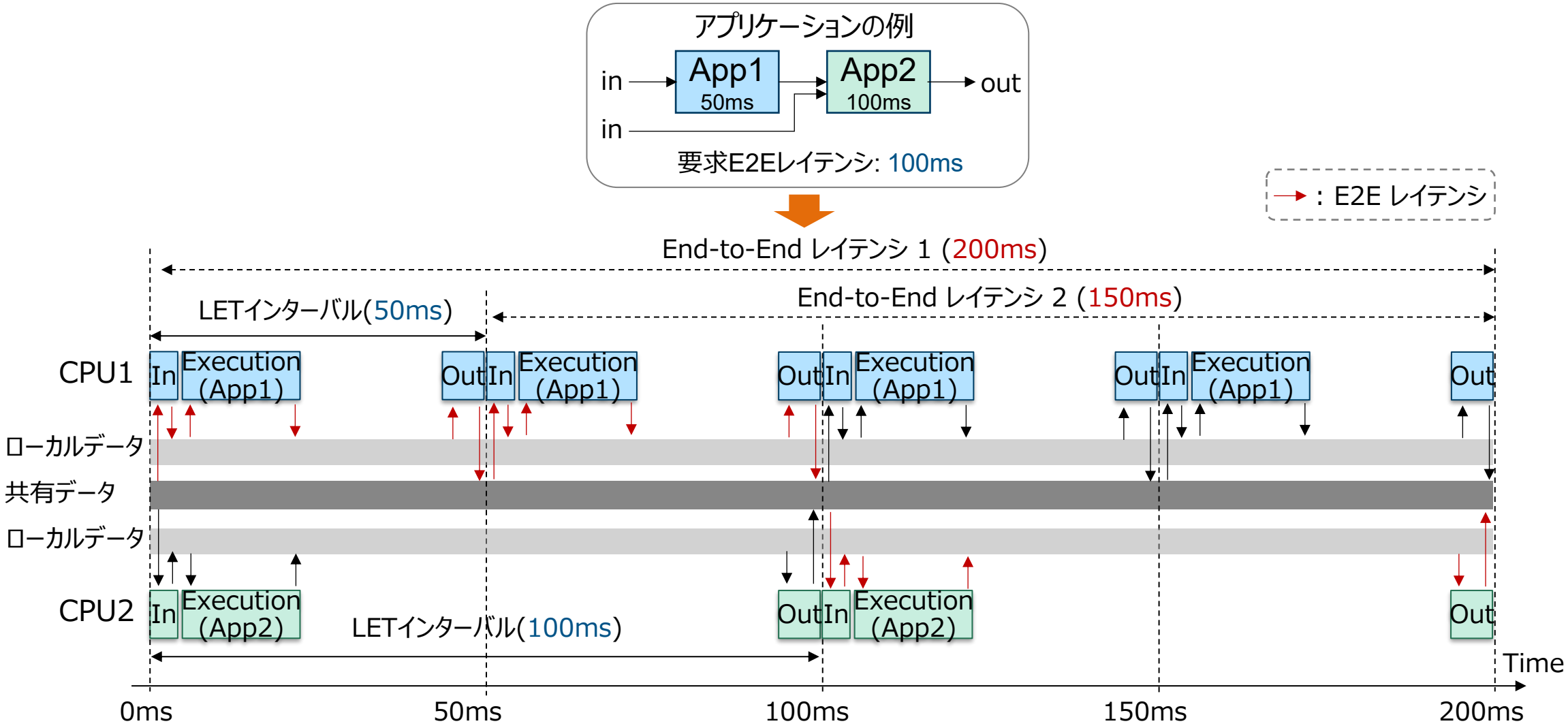
LET Design

LET設計の特徴

- アプリケーションを入力処理(In)、計算処理(Execution)、出力処理(Out)に分離する
- アプリケーション内のデータ通信にはローカルデータを、アプリケーション間のデータ通信には共有データを用いる
- LETインターバル(周期)の最初に共有データを参照(Copy-in)し、最後に更新(Copy-out)する。

従来手法の課題：LET適用時の課題

■ LETを適用するとApp間のデータ通信発生毎に1周期の時間を要するため、レイテンシが長くなる課題がある



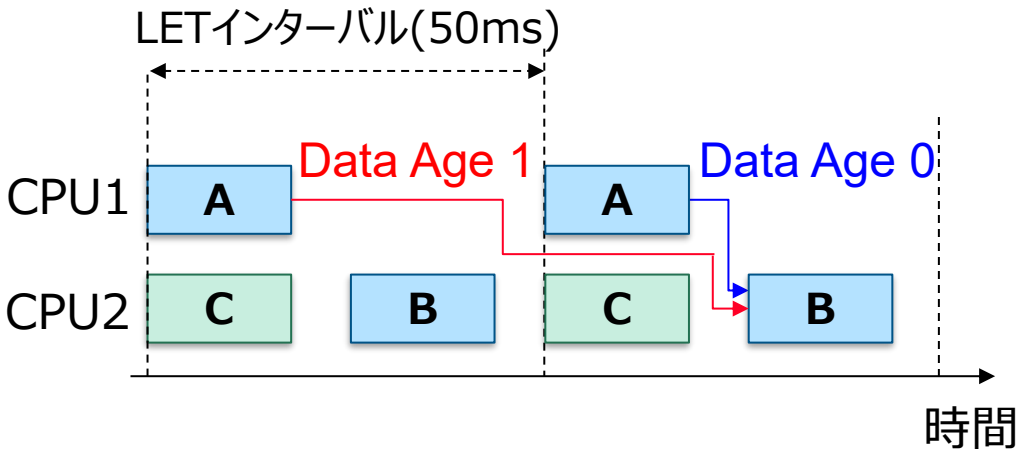
Flexible LET [4]

Astemo
Mobility Beyond

■ LETのデータ通信をData Ageベースに拡張。LET同様参照と最新値参照を選択可能することでレイテンシ短縮

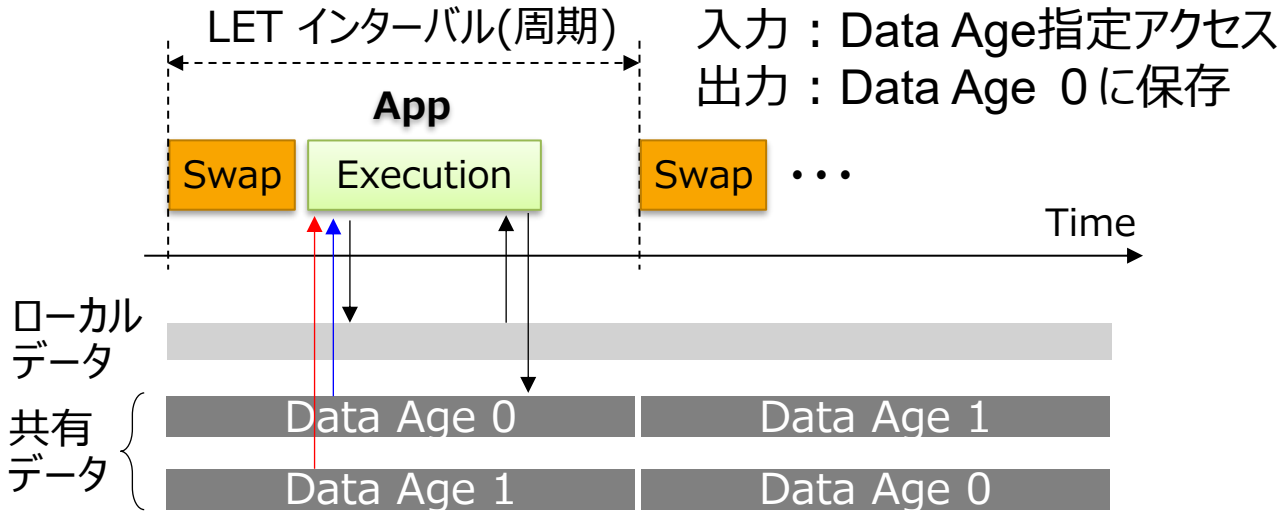
■ Data Age

本研究ではData Ageをデータ通信関係がある
アプリケーション間における送信(出力)と受信
(参照)の想定的な実行タイミングで表す



Data Ageの例

■ Flexible LETの概要



- アプリケーションはData Ageを選択して共有データにアクセス可能
  - Data Age = 0：同じ周期に計算された最新値
  - Data Age = 1：前の周期に計算された前回値(LET同等)
- 周期の最初に各Data Ageが指し示すバッファポインタを変更する
  swap処理を実行することで、LETと同等の「周期の最後に出力
  ＋周期の最初に入力」を実現

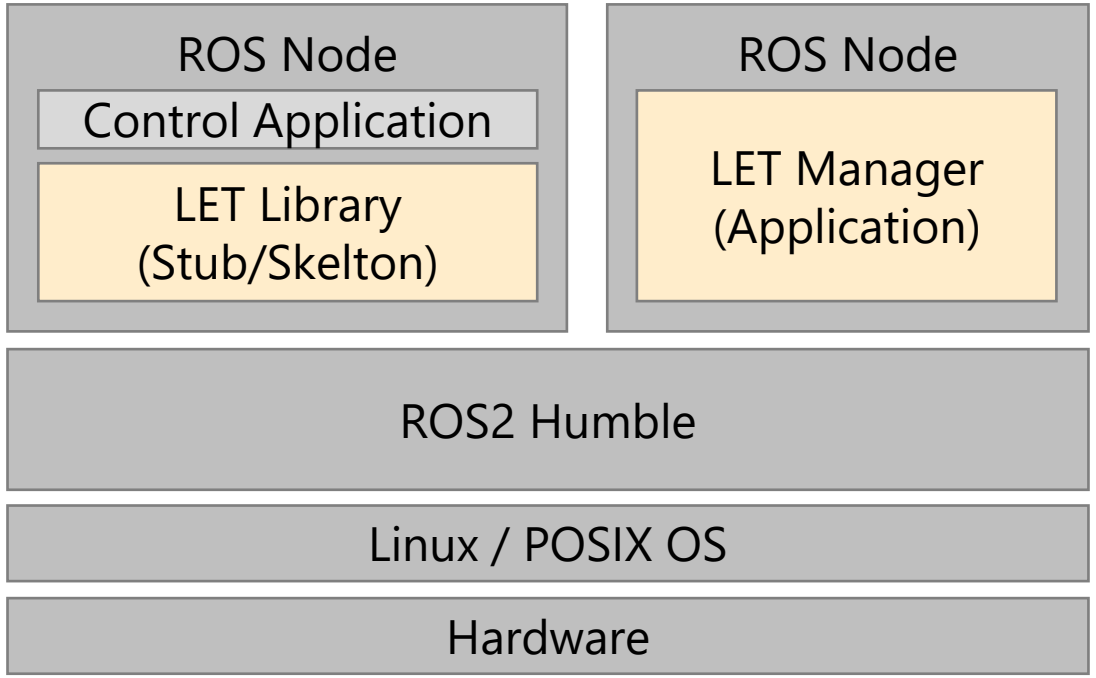
[4]Logical Execution Timeパラダイムの拡張とAUTOSAR Adaptive Platformにおける試作評価、石郷岡他3名、自技会論文誌、2024

ROS2向けソフトウェアスタック

Astemo
Mobility Beyond

■ AUTOSAR APを例題に提案されたFlexible LETをROS2向けに改良し、ライブラリとミドルウェアとして実装

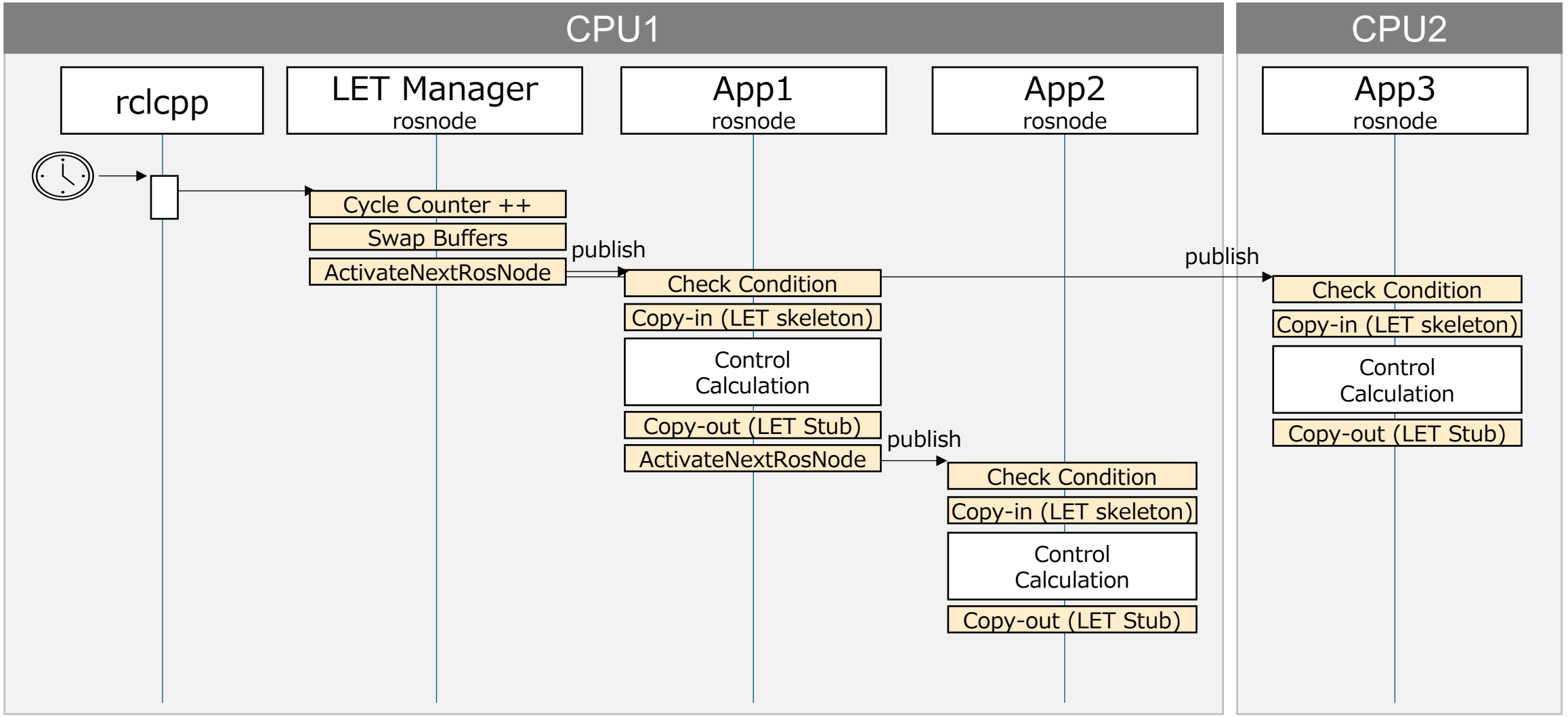
- **LET Manager**： Data Age管理(Swap処理)とアプリケーションの起動を実行
- **LET Library**：アプリケーションにData Age指定の共有データ参照(Skelton)や保存(Stub)のサービスを提供



シーケンス図

Astemo
Mobility Beyond

■ LET Managerは周期的に起動され、実行周期とData Ageの管理(Swap処理)を実施。トピックでAppを起動。



Astemo
Mobility Beyond

Contents

1. 自己紹介、会社紹介
2. Software-Defined Vehicleとクラウドネイティブ開発とは
3. Determinism(決定論)と課題
4. 解決のアプローチとROS2への適用方針
5. 適用評価
6. まとめ

評価環境

Astemo
Mobility Beyond

■ Autowareを例題に、提案手法がcloud-native環境のdeterminismを改善できるか否かを評価する。

■Autowareとは

- ROS2で動作する自動運転OSS
- Autoware Foundationを中心に開発推進

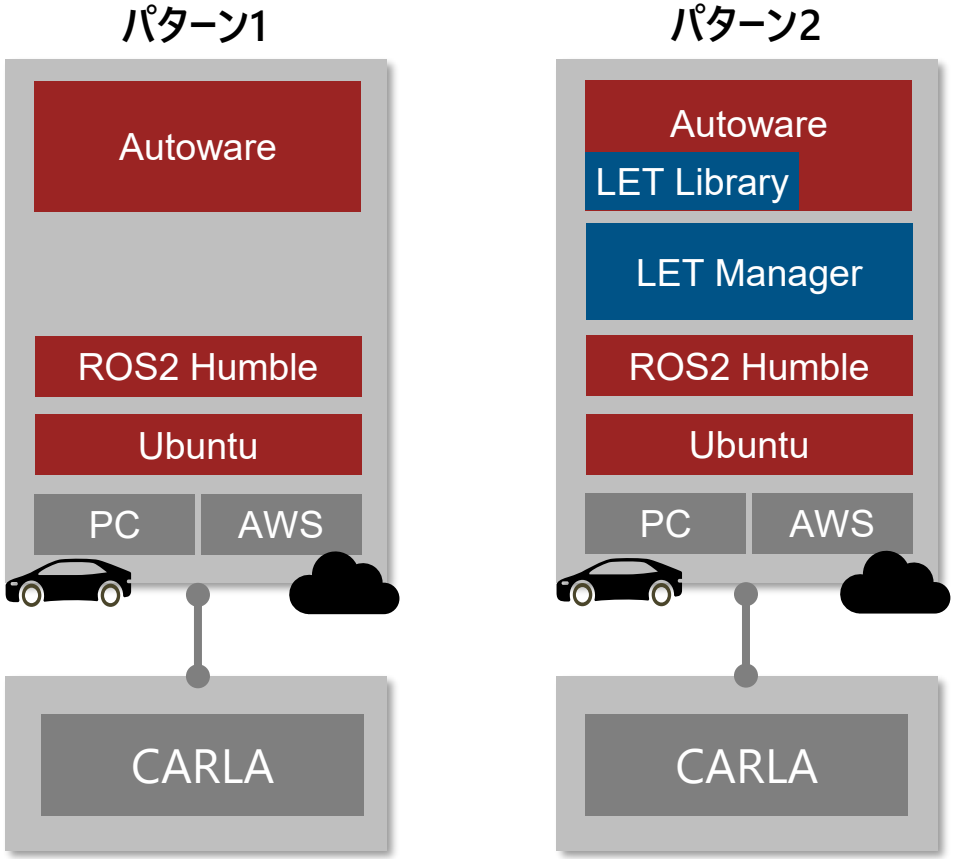
■CARLAとは

- 自動車の走行シミュレータOSS

■評価条件

- Flexible LETをAutowareのPerceptionの一部(localization)に適用
- 意図的に遅延を発生させ、実行時間を変化させる実験を各パターンで10回実施し、走行結果の再現性を比較することでdeterminism改善を評価

■評価パターン

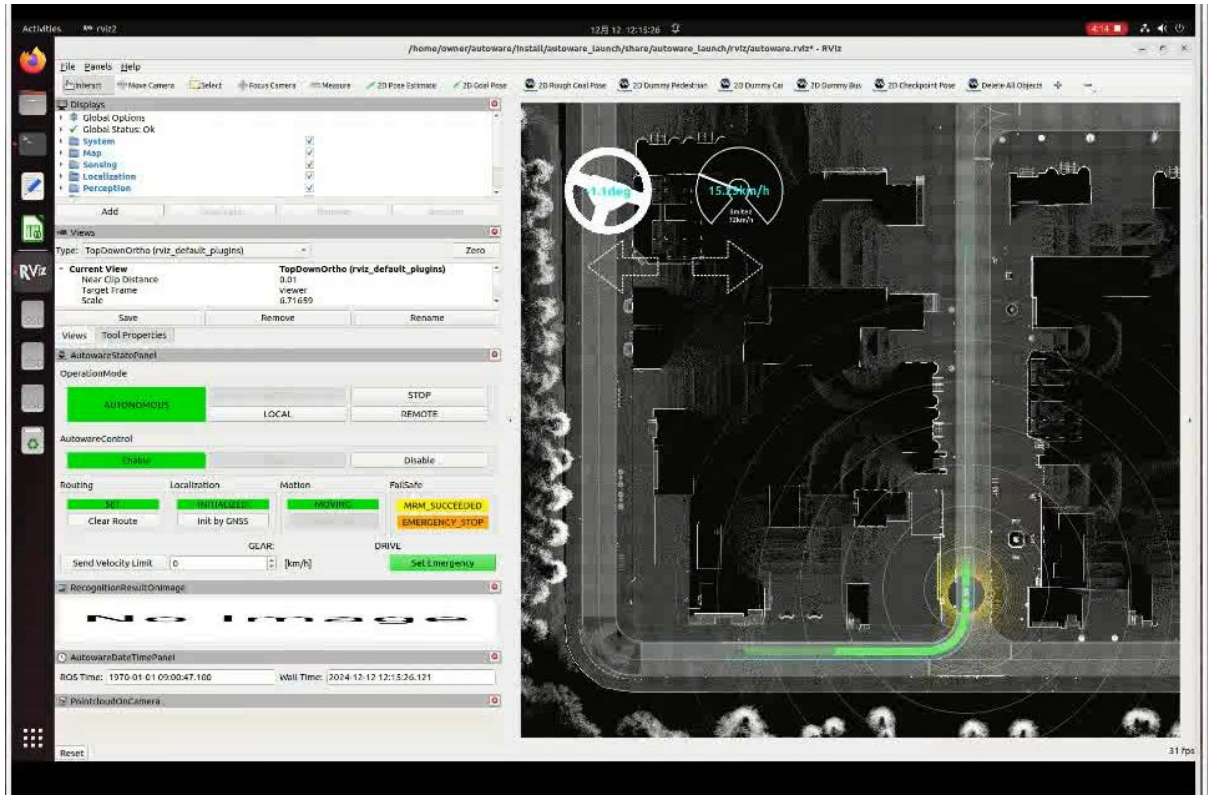


AutowareへのFlexible LET適用結果

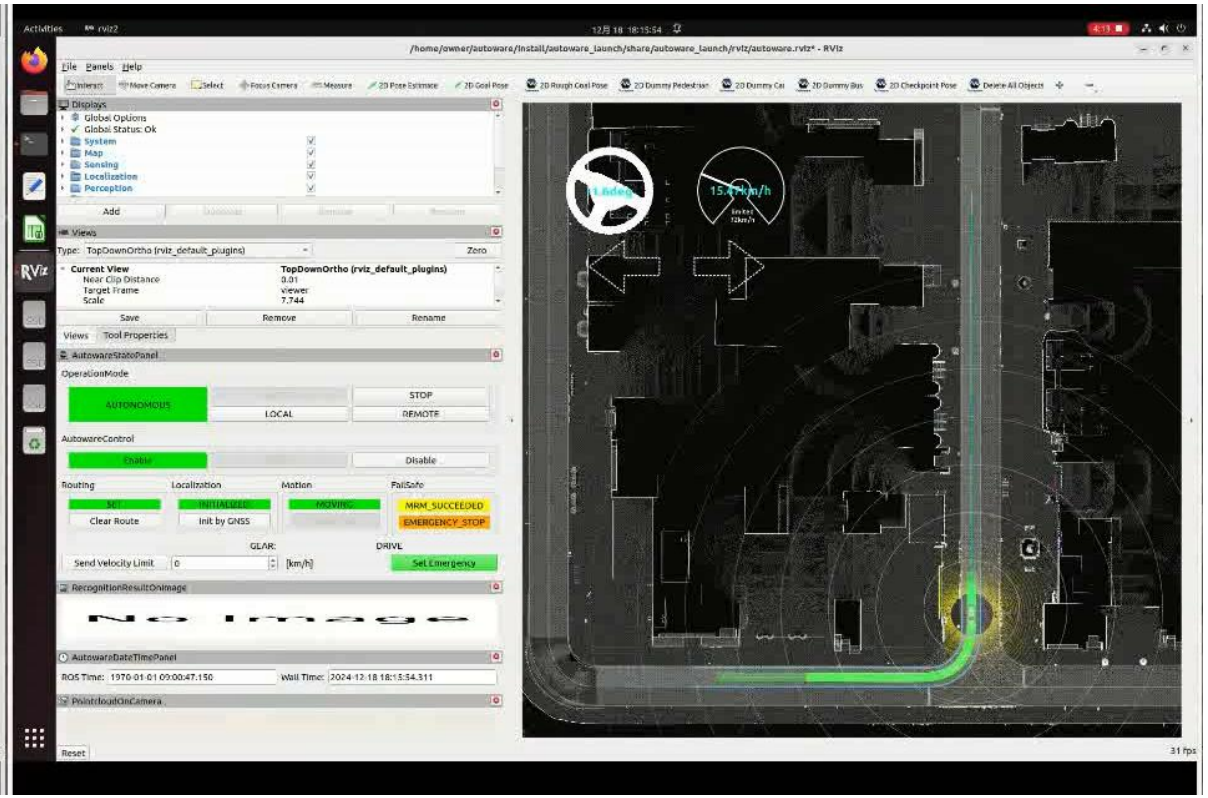
Astemo
Mobility Beyond

- 自動運転OSS(Autoware)をシミュレータで走らせて評価を実施。
- コーナー後の直線で車両を安定させる際に、**適用後のほうが安定化**しているようにみえる。

パターン1のECU/PC (LETなし)



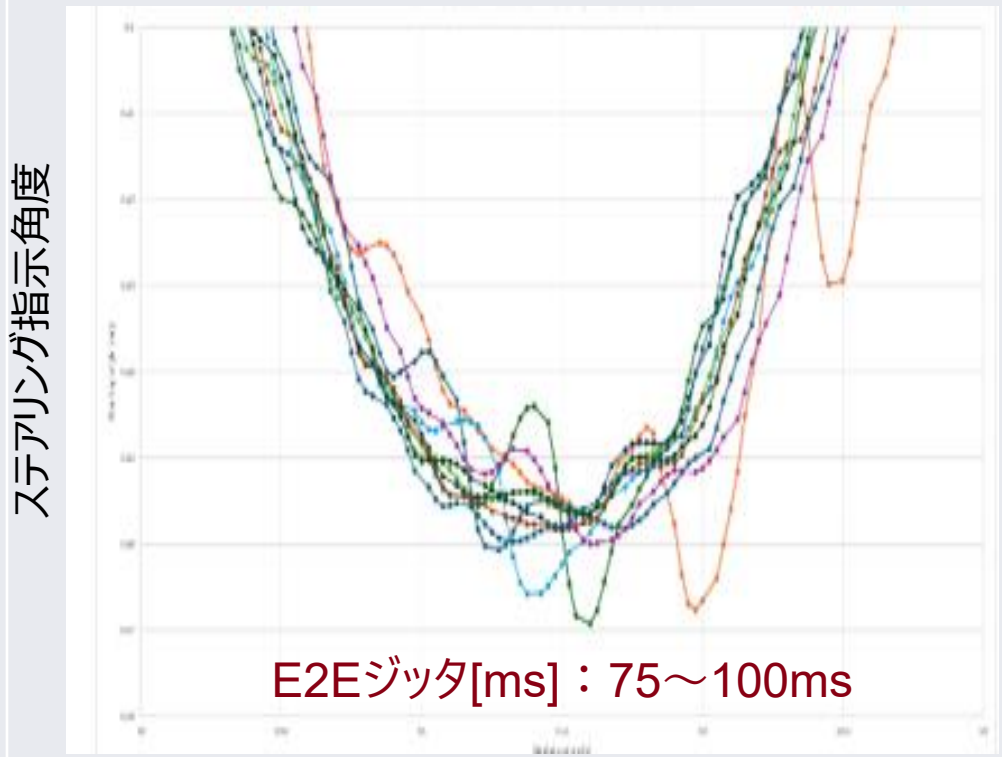
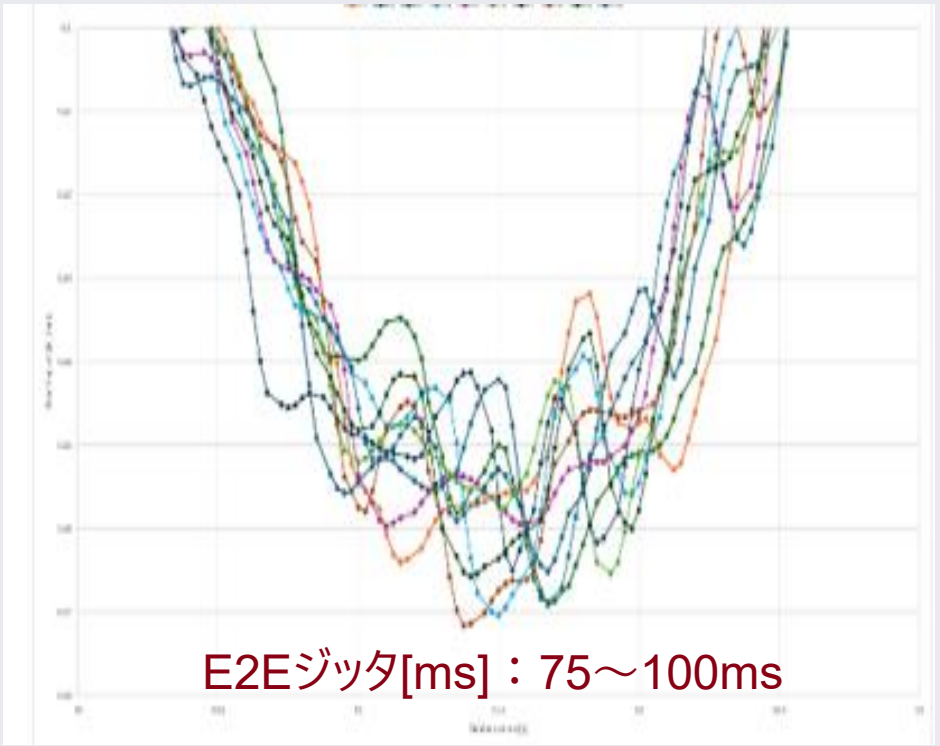
パターン2のECU/PC (LETあり)



Determinism評価結果の詳細：LETなし (1/2)

Astemo
Mobility Beyond

■ **Autowareを用いて、E2Eのジッタとカーブでのステアリング指示角度をグラフ化。AWSはCPU性能が高いためか多少再現性があるが、ECU側は再現性が低い。E2Eジッタが大きい(determinism性が低い)ことが原因と推定。**

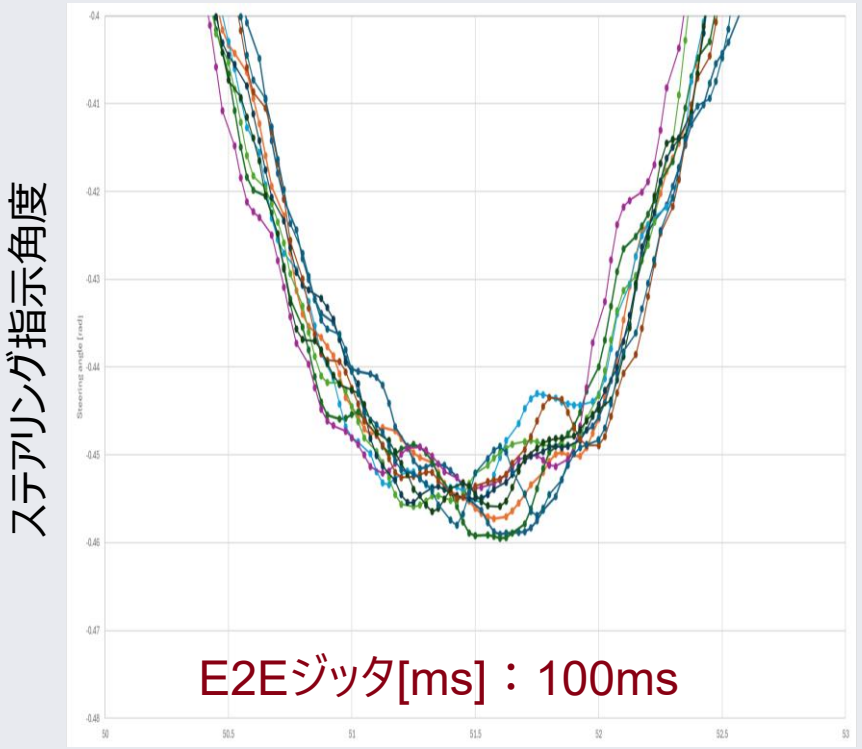
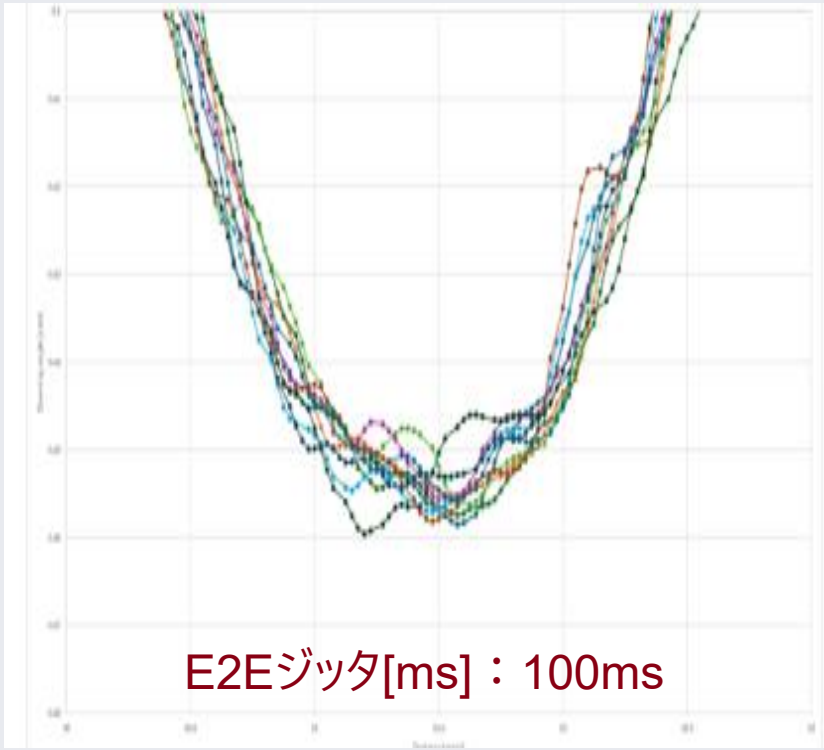
項目	クラウド (今回: AWS)	ECU (今回: PC)
パターン１ (LETなし)		

本評価におけるE2Eは、Flexible LETの適用範囲であるため、センサ入力からPerception/Localizationまでの部分のみとなっている。

※分解能25msのシミュレーション内時間であるため、実態は多少ずれる可能性あり。

Determinism評価結果の詳細：LETあり (2/2)

■ LET適用のほうがクラウド⇔ECU間で**ジッタが改善**されているため、制御レベルでも**再現性が高い**ことがわかる。

項目	クラウド (今回: AWS)	ECU (今回: PC)
パターン2 (LETあり)		

本評価におけるE2Eは、Flexible LETの適用範囲であるため、センサ入力からPerception/Localizationまでの部分のみとなっている。

※分解能25msのシミュレーション内時間であるため、実態は多少ずれる可能性あり。

Astemo
Mobility Beyond

Contents

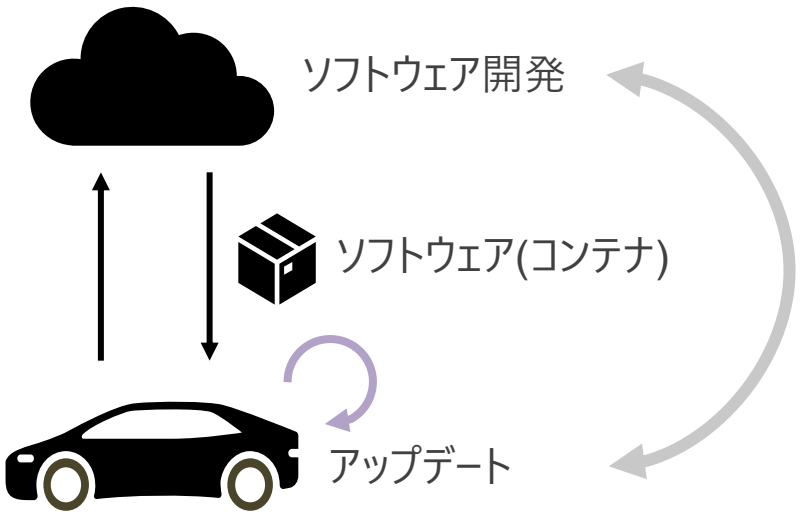
1. 自己紹介、会社紹介
2. Software-Defined Vehicleとクラウドネイティブ開発とは
3. Determinism(決定論)と課題
4. 解決のアプローチとROS2への適用方針
5. 適用評価
6. まとめ

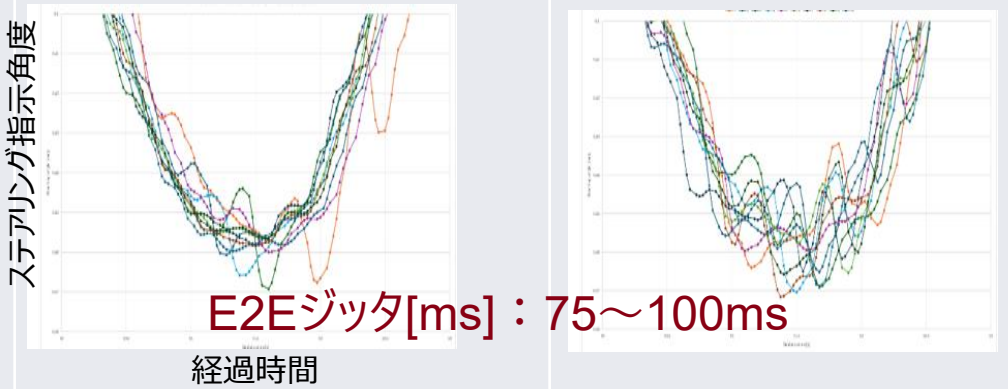
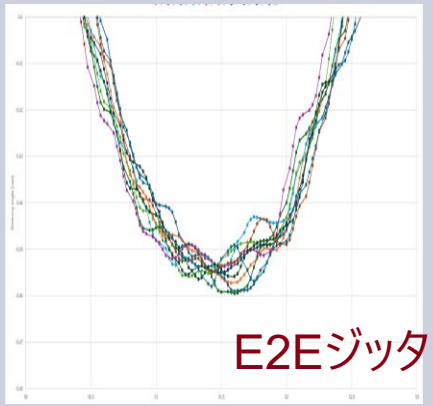
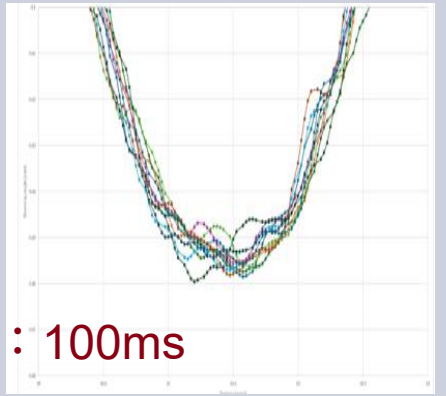
まとめ

Astemo
Mobility Beyond

■ Software-Defined Vehicle 向けのクラウドネイティブ開発環境向けに、Flexible LETをROS2向けに実装
■ Autowareを例題として実験を行い、Determinismを改善可能な見込みを得た

クラウドネイティブ開発の狙い



項目	クラウド (今回:AWS)	ECU (今回:PC)
パターン1 (LETなし)		
パターン2 (LETあり)		

Astemo

Mobility Beyond