# SOAFEE アーキテクチャによるクラウドネイティブな開発の促進ミックスドクリティカルな車載システムへ

Matt Spencer、Arm プリンシパルソフトウェアアーキテクト

ホワイトペーパー

先進運転支援システム（ADAS）、自動運転（AD）、車載インフォテイメント（IVI）など、現代の車載プラットフォームの機能が急速に増加するにつれ、自動車メーカーはソフトウェア定義型実装への移行に注目しています。

この移行は市場の未来を開く鍵であり、コスト削減による利益率の向上、新しい収益源など、多くのチャンスをもたらします。開発/統合コストの増大は、車内の機能ブロックの統合によって抑えることが可能です。異なる車種や世代でコードを再利用できれば、ソフトウェアへの初期投資を償却しやすくなります。

車載分野で使用されるような組み込みシステムのコード開発では、ソフトウェアが、選択したプロセッサーに付属の BSP に含まれる API で記述されることが課題となっています。根本となる BSP に含まれる特定の API に依存しているため、別のプロセッサーへのアプリケーションコードのポータビリティが保証されないからです。

本書では、ソフトウェアのポータビリティ/コンポーザビリティの課題に対して、Arm と車載業界の大手テクノロジーパートナーが提供するソリューションをご紹介します。

## はじめに

多くの機能コンポーネントで構成される複雑なソフトウェアソリューションをセキュアかつ管理された方法で提供したい場合、最初に考えるのはクラウド上で動作する大規模なアプリケーションです。インフラ市場とクラウドサービスプロバイダー（CSP）は、ソフトウェア開発に最善の「クラウドネイティブ」ソリューションを導入し、複雑性を抑えてクオリティを上げるワークフローやツールを構築することで、複雑なソフトウェア導入の問題に対処してきました。

クラウドネイティブ環境には、アプリケーションの開発、導入、ライブ更新の複雑性を軽減するため、導入すべき多数のテクノロジー、ワークフロー、デザインパターンが決まっています。

SOAFEE プロジェクトの目的は、クラウドネイティブ開発環境のメリットを、機能安全（FuSa）や高速かつ正確なリアルタイムコントロールなど、車載分野に特有の課題や制約への対処に活用することです。

クラウドネイティブの基本的な要件の 1 つは、ハードウェアからソフトウェアを切り離せることです。これにより、根本にあるソフトウェアをゼロから作り直さなくても、異なるハードウェアでワークロードを簡単に導入できます。理想は、アプリケーションコードを再コンパイルしなくてもバイナリポーティングできることです。

## クラウドネイティブを車載分野に応用する方法

Cloud-Native Computing Foundation（CNCF）とは、クラウドネイティブのデプロイメントで使用される多くのツールの仕様や導入を管理するオープンソースの財団です。コミュニティのメンバーは、以下のクラウドネイティブの定義を共有し、合意しています：

*「クラウドネイティブ技術は、パブリック、プライベート、ハイブリッドクラウドなど、現代のダイナミックな環境でスケーラブルなアプリケーションを構築および実行する能力を組織にもたらします。このアプローチの代表例に、コンテナ、サービスメッシュ、マイクロサービス、イミュータブル・インフラストラクチャ、宣言型 API があります。*

*これらの手法により、回復力に優れ、管理と観察の容易な疎結合システムを実現します。これらを堅牢な自動化機能と組み合わせることで、エンジニアはインパクトのある変更を最小限の労力で頻繁かつ予測通りに行うことができます。*

*Cloud-Native Computing Foundation は、ベンダーに依存しないオープンソースのエコシステムを育成および維持することで、この技術の普及を推進します。また、最先端のパターンを民主化し、これらのイノベーションを誰もが利用できるようにします」*

この定義からわかるように、クラウドネイティブソリューションをクラウドに展開することを強制しているわけではありません。むしろ最先端のデザインパターンに基づいたコンテナ、マイクロサービス、宣言型 API などの技術の利用を奨励しています。

このような目標は、車載分野における Arm の目標とほぼ一致しています。以下のセクションでは、いくつかの技術と SOAFEE の目標との関係を詳しく説明します。

## SOAFEE は OCI 対応コンテナを使用

Google が規定するコンテナの定義：*「コンテナとは、アプリケーション コードに、ソフトウェア サービスの実行に必要な特定バージョンのプログラミング言語ランタイムやライブラリなどの依存関係を加えた軽量のパッケージを指します」*

ですから基本的に、コンテナとはアプリケーションをパッケージ化して展開する便利な方法です。コンテナ環境は Open Container Initiative（OCI）によって定義され、2 つの主要部分で構成されます。3 つ目の部分では、コンテナレジストリ（hub.docker.com など）との通信に使用する標準規格を表現します。

- Container Runtime Specification
- Container Image Specification
- Container Distribution Specification

この Runtime Specification は、コンテナをシステム上で正常に動作させるために必要なシステム要件とインタフェースを規定します。Image Specification は、コンテナランタイムに許容されるためのイメージ作成方法を定義します。

コンテナのエコシステムにおける標準規格ベースのアプローチのメリットは、コンテナランライム実装に関する革新を奨励し、特定のデプロイメントで分野特有の要件を満たすソリューション開発を促進することです。OCI には runc というコンテナランライムのリファレンス実装がありますが、他にもさまざまな分野に対応する多くのコンテナランライムがあります。

以下に主なコンテナランタイムの例を挙げます：

| ランタイム | 特性 |
| --- | --- |
| runc | go-lang で実装された OCI のリファレンスランタイム |
| crun | C で実装されたフットプリントの小さな軽量ランタイム |
| gvisor | システム API アクセスを制限し、サンドボックス化した高セキュリティのランタイム |
| kata | KVM を使用して仮想化し、セキュリティと分離を改善したランタイム |
| runx | Xen をベースとする仮想化ランタイム |
| ... | その他のコンテナランタイム |

重要なのは、どのコンテナランタイムにも同じソリューションが適しているとは言えないことです。ただし、OCI コンテナ仕様に従って作成されたコンテナであれば、お使いのプラットフォームで修正なしに動作します。

SOAFEE は、アプリケーションとランタイムの強力な分離機能を利用し、車載分野の厳しいデプロイメント特性を満たすコンテナランタイムを存在させるとともに、OCI 内のアップストリームの標準規格がそれらの要件を適宜表現できるようにします。

## マイクロサービス

マイクロサービスアーキテクチャとはソフトウェアデザインパターンであり、疎結合した協調性のあるサービスを確実に作成させ、複数のサービスを一緒に作成することで機能的ソリューションを実現します。このようなサービスには明確に定義されたインバウンドとアウトバウンドのインタフェースがあり、それがサービスとシステム内の他のコンポーネントとのコントラクトを作っています。

クラウドネイティブのデプロイメントではマイクロサービスはコンテナにカプセル化されます。これによりマイクロサービスは定義済みのコンテナランタイム環境内で実行され、デプロイメントはオーケストレーターによって管理およびモニターされます（詳細は後述）。

マイクロサービスが「疎結合した」と定義されるのは、インバウンド/アウトバウンド API のコントラクトに定義された挙動が守られている限り、1 つのサービスへの変更がシステム内の別のサービスのパフォーマンスに影響を与えないためです。この特性により、マイクロサービスはシステムの残りの部分とは隔離してテストすることが可能です。つまり、大型の複雑なシステムを小さなユニットに分解し、各サービスをテストしてから、完全に組み立てたシステムの統合テストを実行できます。

## オーケストレーター

オーケストレーターはクラウドネイティブシステムの必須要素であり、マイクロサービスベースのソリューションのコンフィギュレーション、デプロイメント、モニタリングを管理します。オーケストレーター自体は複数の標準インタフェースで構成されています：

| インタフェース | 概要 |
|---|---|
| CRI | オーケストレーターとコンテナランタイム間のインタフェース |
| CNI | コンテナネットワークインタフェース：ネットワークやファイアウォールなどを設定または制御する標準メカニズム |
| CSI | コンテナストレージインタフェース：コンテナインスタンスでストレージを使用可能にする方法 |
| デバイスプラグイン | /dev/video0 などのコンテナ内でシステムリソースへのマネージドアクセスを有効化 |
| … | 他の標準規格が適用される場合あり |

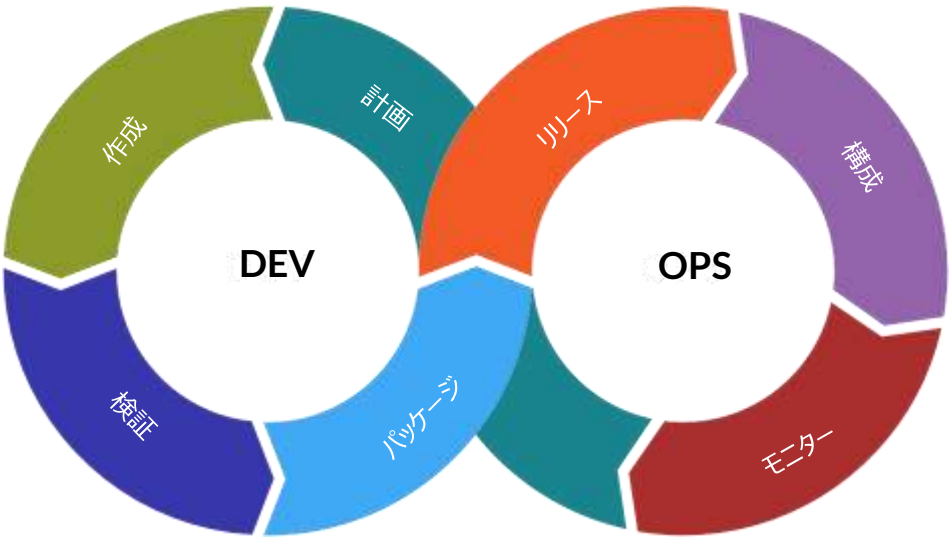
クラウドネイティブエコシステムの他のすべての側面と同様、特定用途のニーズを満たす標準インタフェースの実装はいくつかあります。オーケストレーターでコンテナランタイムを管理したい場合は CRI インタフェースを実装する必要があり、本書でこれまでに説明したランタイムはすべてその要件を満たしています。

これらのインタフェースをオーケストレーターでまとめると、マイクロサービス間の通信や、正常に機能するために必要なデータソースへのアクセスを有効化するようネットワークを構成し、複雑なアプリケーションデプロイメントを管理できます。

オーケストレーターには多くのオプションがあります。デフォルトは kubernetes（k8s）ですが、組み込み環境やリソースの少ない環境に適したフットプリントの小さい k3s などの実装もあります。

## DevOps

クラウドネイティブのワークフローは一般に DevOps プロセスと呼ばれます。




*出典：commons.wikimedia.org/wiki/File:Devops-toolchain.svg*

ワークフローは主に 2 つの部分に分かれます。Dev とは開発のワークフロー、Ops とはデプロイメントの運用の側面です。2 つの部分を明確に定義/管理された形で組み合わせれば、このワークフローで管理されるアプリケーションの開発、導入、継続的な改善を合理的に実行できます。

上の図からわかるように、これは継続的なプロセスであり、デプロイメント内のコンテナの運用をモニタリングした結果が次の開発サイクルに活用されます。

こうして、実行されるワークロードの質は継続的に改善されます。時間が経つにつれ、このプロセスが全体のクオリティを高めるとともに、製品開発期間の短縮とコスト削減に貢献します。

# SOAFEE がクラウドネイティブを促進

SOAFEE プロジェクトは、クラウドネイティブの枠組みを通じて、使用されているベストプラクティスや標準規格を活用します。課題は、車載ソリューションには他の要件や制約があることです。たとえば、アプリケーションプロセッサー、リアルタイムプロセッサー、各種のアクセラレーターが混在するヘテロジニアス処理アーキテクチャにワークロードを導入できることです。

SOAFEE は、SOAFEE ワーキンググループを通じてクラウドネイティブ実装に現在欠けている点を理解し、関連の標準化団体内でアップストリームに向けて取り組みます。欠けている点を共同で埋めることにより、クラウドネイティブソリューションは、車載分野や安全関連の分野にも適用可能となります。

### オーケストレーターの改良 – 安全性とリアルタイム性

オーケストレーターのスケジューリングフレームワークでは、一部の側面で、以下のような標準的なテクニックで安全性とリアルタイム性の要件に対処する必要があります。

- ノードアフィニティ：ワークロードをデプロイする場所を制限
- Taints と Tolerations：特定のノード同士がどのように引き付け合い、あるいは排除し合うかを記述
- Pod オーバーヘッド：特定のワークロードが消費するシステムリソースの量を記述

しかし現在のシステムには欠点があります。SOAFEE は標準化団体とアップストリームに向けて取り組み、リアルタイム性と安全性の要件を記述する言語を正規化する予定です。

例 – リアルタイム性の要件

- 必要な I/O 帯域幅
- 実行時間の保証
- キャッシュポリシー

例 – 安全性の要件

- 無干渉
- Split-lock コア
- 可用性

上記の拡張分野については、それぞれ SOAFEE テクニカルワーキンググループが、オーケストレーターに意図を表現する共通の言語の作成に取り組む予定です。これによりワークロードを基本システムの適切な部分にデプロイし、低レベルのアーキテクチャ機能を各 Pod 向けにコンフィギュレーションすることで要件を確実に満たします。

### Container Runtime 拡張

オーケストレーターがワークロードの付加的なランタイム要件を適切に表現できるようになったら、ニーズに合わせてコンテナランタイムを拡張します。第一のパスプロポーザルは、前述の runx のような仮想化されたコンテナランタイムを使用し、ランタイム自体とともに VMM の使用を可能にすることです。これにより VMM を通じた特権システムリソースの制御を、コンテナランタイムの低いほうの特権実行環境から分離します。

SOAFEE ワーキンググループは、正しい初期実行環境を選択し、修正が必要な場合には、OCI の標準化団体とアップストリームに向けて取り組みます。また、選択したランタイムとも協力して拡張を実装します。

### ポーティング可能なワークロード

SOAFEE の重要な価値提案はワークロードの再利用です。これにより、最終的なアプリケーションを構成する具体的なマイクロサービスを複数の製品ラインやソリューションに修正なしで再利用し、複雑なソフトウェアソリューションの導入コストを削減できます。

再利用を可能にするには、アクセラレーターや IO デバイスの特定アーキテクチャ向け実装に合わせてリコンパイルすることなく、アクセラレーターや広帯域幅の IO デバイスにワークロードへの安定したアクセスを与える方法を理解する必要があります。このコンセプトを探るには、機能安全とリアルタイム性に関する分野特有の要件を念頭に置く必要があります。

まず頭に浮かぶ業界標準が VirtIO です。VirtIO は、アクセラレーターへの準仮想化したアクセスを提供し、事実上、アクセラレーターのワークロードビューを標準化するとともに、バックエンドアクセラレーターへの効率的なオフロードを可能にします。

一見、VirtIO はポータビリティの問題への完璧なソリューションのように見えますが、現行リリースの VirtIO には課題があります。1 つは、機能安全やリアルタイムワークロードを想定して設計されていないことです。もう 1 つは、インタフェースが車載分野のニーズすべてに対応しないことです。たとえば TVM などのユーザースペースを通じて機械学習アクセラレーションに対応する VirtIO インタフェースはありません。CAN バスのような車載分野に特有の IO デバイスへの標準インタフェースもありません。

SOAFEE は、エコシステムの断片化を招きかねない新しい標準規格を作るのではなく、既存の標準規格の採用を優先し、それらを機能安全分野の目的に適応させます。また、VirtIO の標準化団体内でアップストリームに取り組み、問題解決に努めます。これには、virtqueue でリアルタイム制約を表現できるようにして問題を回避することや、足りないインタフェースに関して妥当な VirtIO 実装を定義する業界内の協働を確保することが含まれます。

## テストと検証

マイクロサービスベースのソリューション開発から生じる大きな可能性の 1 つは、コンポーネントレベルとシステムレベルでのテストと検証に対応する CI/CD ツールを実現することです。たとえば SOAFEE のワークロードポータビリティ機能を利用すれば、クラウド内でワークロードトレーニング/テストを実行し、ワークロードパフォーマンスにおける第 1 レベルの信頼性が得られます。次に、この同じワークロードをラボベースのインフラに導入すれば、ソフトウェアとハードウェアの両方をループ検証に応用できます。

前述の標準的な DevOps ワークフローでは、クラウドネイティブのデプロイメントが複雑なワークロードのクオリティをどのように維持するかを示しました。SOAFEE はまさにこのワークフローの実行と改良を可能にします。SOAFEE プロジェクトは、システムインテグレーター、ツールベンダー、CSP と協力し、その実現に努めています。

SOAFEE のこの重要な側面については、今後さらに詳細をご紹介する予定です。

## オープンソースのリファレンス実装

この取り組みすべての成果が、SOAFEE の要件のオープンソースリファレンス実装です。これにはプロジェクトが目指すクラウドネイティブの実現に向けて必要な要素がすべて含まれます。リファレンスは、ベースプラットフォームを別のハードウェアにポーティングするための Yocto レシピの形で提供されます。

次に、このリファレンス実装を Autoware、AGL などのアップストリーム豊富なソフトウェアスタックで使用すれば、本当にポーティング可能でコンテナ化されたマイクロサービスベースの実装が可能となります。これは SOAFEE アーキテクチャを実装したあらゆるプラットフォームで動作します。

現行の SOAFEE スタックの構築方法および対応ハードウェアの詳細は、SOAFEE プロジェクトページをご覧ください。詳細はこちら：gitlab.arm.com/soafee

## まとめ

SOAFEE プロジェクトでは、自動車メーカー、大手半導体企業、大手クラウドテクノロジー企業を集め、ソフトウェア定義型車両向けの新しいオープン標準ベースのアーキテクチャを定義しています。SOAFEE は、コンテナオーケストレーションなどのクラウドコンセプトを車載機能安全と組み合わせるためのリファレンス実装を業界で初めて提供します。その土台となるのは、Arm アーキテクチャの標準的なブートとセキュリティの要件を定義した Arm の取り組み、Project Cassini の成功です。

クラウドネイティブ技術は、ソフトウェアとハードウェアのループなど CI/CD の高度な手法のほか、トレーニングや検証におけるクラウドベースのインフラの利用を可能にします。

SOAFEE は、ソフトウェア定義型車両の複雑性および開発や導入のコスト削減に貢献します。また、インテグレーションをやり直すことなく既存のワークロードを新しいアーキテクチャで運用可能にすることで、エコシステムによるソフトウェア投資の再利用を最大限に促進します。

詳細：Arm のウェブサイトをご覧ください
お問い合わせ：こちらのフォームをご利用ください

## 用語集

| 用語 | 意味 |
|---|---|
| AD | 自動運転 |
| ADAS | 先進運転支援システム |
| AGL | Automotive Grade Linux |
| API | アプリケーション・プログラミング・インタフェース |
| BSP | ボードサポートパッケージ |
| CI/CD | 継続的インテグレーション/継続的デプロイメント |
| CNCF | Cloud-Native Computing Foundation |
| CSP | クラウドサービスプロバイダー |
| IVI | 車載インフォテイメント |
| OCI | Open Container Initiative |