4A-05

組込機器へのコンテナ技術適用における一考察

宮田　幸太†　菊田　祐司†　宮崎　剛†

富士電機株式会社†

1 はじめに

近年，企業における IT システムの役割は,利便性を求めるものから,IT 技術を利活用し新たなビジネスを創出するものへと変化している.上述の環境では,実現したい要件を模索しながら開発を行う技術や変化する顧客要求を素早く開発にフィードバックする手法が求められており,マイクロサービスやハイブリッドクラウドといったソフトウェアのプラットフォームが台頭している[1].

富士電機においても工場設備，ライン，工場全体のエネルギー管理，生産性向上，予知保全など IoT を活用した新たなサービスと価値の提供を進めている．また，産業用コントローラの分野では AI・解析・診断などの付加価値サービスの増加が一層見込まれる.更に,自販機などの小売領域では，海外からのインバウンド増加など社会ニーズに合わせたコンテンツが必要となり，アプリの開発・更新が容易にできる仕組みが必要となってきている.

このように，産業分野においてもサービスの迅速な更新や展開が必要とされ,組込機器を含む産業システムにおいてもソフトウェアの独立性，再利用性を向上させる技術開発が必須となってきている.

本稿では上述の技術開発の一環として取り組んできた産業用コントローラにおけるマイクロサービス化の検証,技術適用に向けた考察について述べる.

2 富士電機での取り組み

我々は産業システムのマイクロサービス化に向けた基盤技術として,コンテナ技術の検証に取り組んできた.具体的には 図 1 に示すようにクラウドやオンプレミスだけでなく,産業用コントローラのような組込機器を含めたシステムへのコンテナ適用検証やクラウドベンダーが提供するマネージドサービスを用いたサービス検証,マイクロサービス管理ツールの検証を実施してきた.

3 適用課題

産業用コントローラではハードウェア制約があるため,必要なストレージ・リソース使用量等が課題となる.また,適用に際して ARM,X86 など異なる CPU アーキテクチャを想定する必要がある.本稿では以下の 3 項目に分け,産業用コントローラへの適用検証について述べる.

① 汎用ボードでの適用検証
② 自社産業用コントローラでの適用検証
③ 産業用コントローラ適用への考察

A study on the application of container technology to embedded devices

Kota Miyata†, Yuji Kikuta†, Tsuyoshi Miyazaki†
†Fuji Electric Co.Ltd

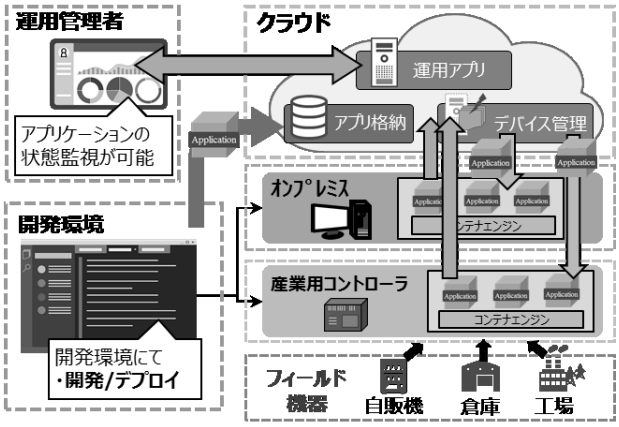

図 1　マイクロサービス適用検証概略

4 検証

4.1 汎用ボードでの適用検証

産業用コントローラへの適用可否を検証するため,異なる CPU アーキテクチャの汎用ボードに対しコンテナエンジン(Docker[2],Moby[3])を実装し,コンテナ技術の原理確認,コンテナ環境における性能評価,構築に必要なリソースの計測を実施した.今回,対象とした X86 系チップのデバイス,ARM 系チップのデバイスのスペックを表 1 に示す.

検証結果として,各 CPU アーキテクチャデバイスにおける基本動作を確認し，コンテナ上においてもホスト OS と同等の演算性能が得られることが分かった.また,検証を通してアーキテクチャによるコンテナイメージの構成差異を確認した.更に,表 2 に示すように各コンテナエンジン実装における必要リソースを確認し,実装可能な自社産業用コントローラの抽出を行った.

表 1　汎用ボード検証環境

デバイス	X86 系デバイス	ARM 系デバイス
ホスト OS	Linux 4.4.0	Linux 4.9.3
バージョン	Ubuntu 16.04.5	Raspbian
CPU	1.2GHz クアッドコア VIA Eden X4	1.2GHz クアッドコア Cortex-A53
RAM	2GB	1GB
ROM	32GB	16GB

表 2　コンテナ実装における必要リソース

コンテナエンジン	Docker V19.03	Moby V3.0.6
CPU	シングルコア 300Hz 以上	シングルコア 300Hz 以上
メモリ	100MB 以上の空き	80MB以上の空き
ストレージ	420MB 以上の空き	260MB 以上の空き
OS	LinuxKernelv3.10 以上	LinuxKernelv3.10 以上

4.2 自社産業用コントローラでの適用検証

次に自社の産業用コントローラ上での検証を実施した. 具体的には,選定した自社産業用コントローラに前述のコンテナエンジン(Docker,Moby)を実装し,自社の解析診断アプリを動作させた.その上で,リソース（CPU,メモリ,ストレージ）使用量および処理時間（解析診断アプリの目標性能：500[ms]以内）を計測し,コンテナ実装によるオーバーヘッドを分析した.今回対象とした自社産業用コントローラのスペックを表3に示す.

各項目の測定結果を表4に示す.図3より,アプリの処理時間はホストと比較してDockerでは28.6[ms],Mobyでは56.4[ms]増加しているが,コンテナ化した自社解析診断アプリでも十分に要件を満たすことが確認できた.単一コンテナであれば,現行エッジでも許容できる範囲のオーバーヘッドであるが,速度が求められる処理や複数コンテナを実行するには,現行ハードではリソース不足が懸念される.そのため,より性能の高いハードや大容量のストレージが必要と考えられる.また,図4,5,6よりMobyコンテナを用いることで処理性能はDockerコンテナと比較して低下するが,ストレージ使用量,使用リソース(CPU,メモリ)が低減されることが確認できた.

表3　自社産業用コントローラ検証環境

デバイス	ARM系自社コントローラ
ホストOS	Linux 4.9.146-bone12
バージョン	Debian9.5
コンテナエンジン	Docker-CE 19.03.2 Moby 3.0.6
CPU	800MHz AM3356 Arm Cortex-A8
RAM	512MB
ROM	4GB

表4 測定結果

測定項目		ホストOS	Docker	Moby
ストレージ[MB]	アプリ	0.13	0.13	0.13
	コンテナ	0	96.87	97.87
	エンジン	0	320	152
	合計	0.13	417	250
CPU[%]	アプリ	2.8	3.1	2.3
	コンテナ	0	0.5	0.6
	エンジン	0	0.8	0.8
	合計	2.8	4.4	3.7
メモリ[MB]	アプリ	4.58	4.43	4.68
	コンテナ	0	5.56	6.10
	エンジン	0	91.97	68.39
	合計	4.58	101.96	79.17
処理時間[ms]	アプリ	179.5	208.1	235.9

4.3 産業用コントローラ適用への考察

マイクロサービスを実現する手段の１つであるコンテナ技術を産業用コントローラへの適用するためには,表 5 に示す課題・ポイントを考慮する必要がある.

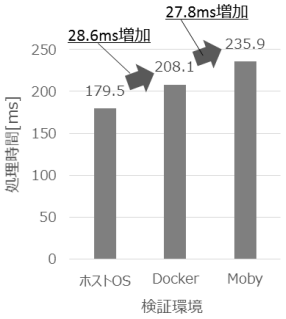

図3　アプリ処理時間

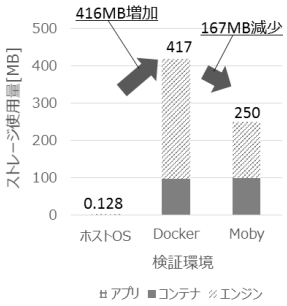

図4 ストレージ使用量

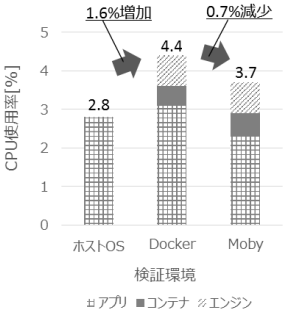

図5　CPU使用率

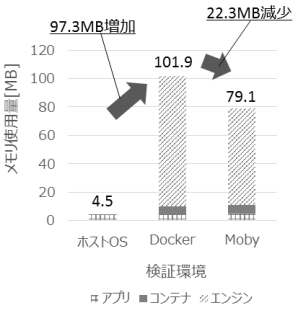

図6　メモリ消費量

表5 産業用コントローラにおけるコンテナ技術適用への課題とポイント

適用について	
1	潤沢なリソースであれば適用可能
	現状の産業用コントローラにて複数のマイクロサービスを用いる場合,ハードウェアリソースが課題となる.
2	実行環境(コントローラ)による課題
	産業用コントローラではレガシー資産活用や顧客要求などから環境を容易に変更できない場合が多い. 適用に向け根本的なシステム変更が必要となる.
運用について	
1	イメージ作成時における注意点
	ARM 用イメージは低機能である場合がある.X86 系と同様に設計してしまうと性能低下の要因となってしまう.
2	異なるアーキテクチャ環境での開発
	ハードウェアのアーキテクチャが異なる場合,コンテナイメージの実行が不可能となる.このような場合でも,エミュレータが搭載された環境を用いることでシームレスな開発・検証を行うことが可能となる.

5 おわりに

本稿では産業用コントローラを含めたシステムのマイクロサービス化を見据え,コンテナ技術の適用検証・考察について述べた.今後,自社産業用コントローラを含めたシステムにおけるマイクロサービス管理技術の検証を進めていく予定である.

参考文献

[1] IPA, 先進的な設計・検証技術の適用事例報告書, https://www.ipa.go.jp/sec/reports/20180228.html

[2] Docker.Inc, Docker, https://www.docker.com/

[3] Moby project, Moby, https://mobyproject.org/