一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
NS2023-47(2023-07)

F-CPS における機能滞留を実現するための WebAssembly を利用したアプリケーション実行基盤の可搬性の性能評価

田中　明弥†　野林　大起††　塚本　和也†††　池永　全志††　関川　柊††††

† 九州工業大学大学院 工学府　〒804–8550 福岡県北九州市戸畑区仙水町 1-1
†† 九州工業大学大学院 工学研究院　〒804–8550 福岡県北九州市戸畑区仙水町 1-1
††† 九州工業大学大学院 情報工学研究院　〒820–8502 福岡県飯塚市川津 680-4
†††† 株式会社 KDDI 総合研究所
E-mail: †tanaka.meiya767@mail.kyutech.jp, ††{nova,ike}@ecs.kyutech.ac.jp, †††tsukamoto@csn.kyutech.ac.jp, ††††sh-sekigawa@kddi.com

あらまし　発生場所や時間に依存する時空間データとそれを処理する機能 (アプリケーション) を対象の地域内に滞留させることで柔軟なデータ利活用を促進するフローティングサイバーフィジカルシステム (Floating-CPS，F-CPS) が提案されている．F-CPS では，地域内の不特定多数のデバイス上で共通の機能を実行する方法として，コンテナ仮想基盤の利用が想定される．しかし，情報滞留システムを用いて原則ブロードキャスト通信を利用する F-CPS において，データサイズが大容量になるコンテナではデータ流通に支障が出る可能性がある．そこで，F-CPS では WebAssembly を利用した小型で汎用的なアプリケーション実行基盤が提案されており，本研究では，実機実験を通してその可搬性を評価する．実験の結果，データ転送時間を最大 97.7% 削減し，F-CPS における高い可搬性があることを確認した．

キーワード　時空間データ，F-CPS，情報滞留システム，機能滞留，コンテナ，WebAssembly

Proposal of Application Execution Platform using WebAssembly for Data Retention System

Meiya TANAKA†, Daiki NOBAYASHI††, Kazuya TSUKAMOTO†††, Takeshi IKENAGA††, and Shu SEKIGAWA††††

† Graduate School of Engineering, Kyushu Institute of Technology
1–1 Sensui-cho, Tobata-ku, Kitakyushushi, Fukuoka, 804–8550 Japan
†† Faculty of Engineering, Kyushu Institute of Technology
1–1 Sensui-cho, Tobata-ku, Kitakyushushi, Fukuoka, 804–8550 Japan
††† Faculty of Computer Science and Sytems Engineering, Kyushu Institute of Technology
680–4, Kawadu, Iizukashi, Fukuoka, 820–8502 Japan
†††† KDDI Research, Inc.
E-mail: †tanaka.meiya767@mail.kyutech.jp, ††{nova,ike}@ecs.kyutech.ac.jp, †††tsukamoto@csn.kyutech.ac.jp, ††††sh-sekigawa@kddi.com

Abstract　A Floating Cyber-Physical System (F-CPS) has proposed to promote flexible data utilization by retaining spatio-temporal data, which depends on the location and time of data generation, and the functions (applications) within the specific area. F-CPS is expected to use container technologies to achieve common functionality across many devices. However, because F-CPS also assumes the direct exchange of functions between devices, conventional container technology with large data sizes may hinder data and function distribution within the area. Therefore, F-CPS proposes a compact and versatile application execution platform using WebAssembly. In this paper, we evaluate its portability through experiments using actual devices. The experimental results show that the data transfer time is reduced by up to 97.7% and the high portability.

Key words　Spatio-Temporal Data，F-CPS，Data Retention System，Application Retention，Container，WebAssembly

1. はじめに

Internet of Things (IoT) やクラウド技術，AI 技術の発展と普及に伴い，現実世界の様々な課題を解決するネットワークのアーキテクチャとしてサイバーフィジカルシステム (Cyber Physical System : CPS) の実現が期待されている．これはスマートフォンや車両，センサ等のデバイスエッジによって収集された現実空間 (フィジカル空間) の情報をサイバー空間に集積・分析し，その分析結果や結果に基づくサービスを現実世界にフィードバックすることで課題解決を目指すシステムである．CPS 実現に向けて，Beyond 5G/6G などの次世代無線通信技術の活用が重要であり，効率的にデータを収集・分析・活用するための新たなネットワーク基盤が必要となる．そこで，我々の研究グループでは，地域特化型の CPS として，特定の地域の時空間情報とそれを処理する機能をその地域に滞留させることが可能であるフローティングサイバーフィジカルシステム (Floating Cyber Physical System : F-CPS) を提案している [1] [2].

F-CPS では，交通情報や気象情報，時限的な店舗広告といったデータの発生場所や時間に依存する「時空間データ (Spatio-Temporal Data : STD)」とそれらを処理・活用するための「機能 (アプリケーション)」を特定の地域内に拡散・維持し，地域内のユーザが所有する通信・計算能力を有したデバイスエッジによって活用することで，サイバー空間と現実空間の連携，融合を目指している．これに対し，F-CPS を実現するためのデータ流通の手段として 時空間データ滞留システムが提案されている [3]- [7]．時空間データ滞留システムは，STD を遠隔地に設置されたサーバを経由せずに，地域内の端末がブロードキャスト通信を用いることによって，直接ユーザに提供する．また，F-CPS では，特定の場所，時間における柔軟なデータの利活用を目的に，ユーザが受け取った STD を有効活用する手段として，STD を処理する機能 (アプリケーション) の滞留が検討されている．この機能の滞留に関しては，不特定多数のデバイスエッジ上において共通のアプリケーションとして実行する必要があるため，ハードウェアやオペレーティングシステム (Operationg System, OS) に依存しないシステムである必要がある. 現状における一般的な実現手段としては，仮想化技術の一つであるコンテナの利用が考えられるが，無線通信を用いてアドホックに特定の地域内のデバイスにアプリケーションを配信する F-CPS において，データサイズが大容量となるコンテナを利用した場合，機能及びデータ流通に支障が出る可能性がある.

そこで，我々の研究グループでは，F-CPS の実現に向けて様々なデバイスエッジで実行可能であり，小型で汎用性の高いアプリケーション実行基盤として，アーキテクチャやカーネルに依存せず実行可能な WebAssembly (Wasm) の活用を検討している [8]. 本稿では，Wasm を活用した実行基盤による「機能滞留システム」を想定し，Wasm を利用した実行基盤と，コンテナ技術を活用した実行基盤の F-CPS における可搬性を，Wi-Fi とローカル 5G を利用した 2 つの実験環境において評価，比較することで検証する.

以下，2 節では関連研究，3 節では F-CPS における機能滞留及び機能実行基盤について述べる．4 節では実機実験における実験結果と評価を示し，最後に，5 節でまとめを述べる.

2. 関連研究

本節では関連研究として，ハードウェアに制約のある IoT デバイスにおけるコンテナとして利用するための，Wasm の適用可能性について調査している研究について記す．コンテナ技術の現状として，Docker による実行・管理が普及している．Docker は高性能であるが，大規模で複雑なシステムであり，データセンタ内のサーバで実行する場合と異なり，ハードウェアに制約のある IoT デバイスでは，システムメモリの消費や起動時間の面での欠点が顕著になる．そこで, [9] では，IoT デバイスのための小型で軽量なコンテナ実行基板として Wasm の利用を提案している．Wasm とは仮想マシン上で実行することを目的とした仮想命令セットアーキテクチャ (Virtual Instruction Set Architecture : Virtual ISA) であり，広義ではバイナリコードを生成し，実行する環境全体を指す．ISA とは，特定のマシンで実行することを目的として設計されるバイナリフォーマットのことであるが，Wasm においては仮想マシン上での実行を目的としており，物理的なハードウェアや OS に依存せず，様々なプラットフォームでの展開が可能である．Wasm が注目される理由として，Chrome や Firefox，Edge などの主要な Web ブラウザに対応しており，プラグインを使うことなく利用可能である点と，Wasm 実行ファイルがバイナリコードであるためデータサイズが小さく，高速に動作するという点がある．そのため，Wasm をクライアント側に導入することで，サーバの負荷やネットワークトラヒックを大幅に削減することが可能となる．Wasm コンテナは単純なランタイムを採用しながら，Docker コンテナとほとんど同じ利点を提供することができ，ハードウェアに制約のある IoT デバイスにも適している.

[9] では，Wasm と Docker コンテナにおいて，IoT デバイスで実行可能なファイル間の定量的な比較実験を通じた様々なパフォーマンス指標の分析を行っている．その結果として Wams コンテナはオーバーヘッドが大きいが，システムメモリの消費と起動時間の短縮により Docker コンテナを上回る性能であると述べている．よって，Wasm コンテナは，単純なプログラムの散発的な実行が必要な場合に最適な選択であり，ソフトウェアの全体ではなく短い機能が使用されるエッジでのサーバレスコンピューティングに適している可能性があると結論づけている．これらの研究では，プログラム実行における定量的な比較，分析が行われているが，データ流通における性能については検討されていない．そこで，本稿では，1 対多のコンテナ配信における性能評価として，従来のコンテナと Wasm コンテナによる比較実験を実施し,Wasm によるアプリケーション実行基盤の可搬性について検証する.

3. F-CPS における機能滞留及び機能実行基盤

本節では，F-CPS において提案している機能滞留システムの目的及びその概要について記述する.

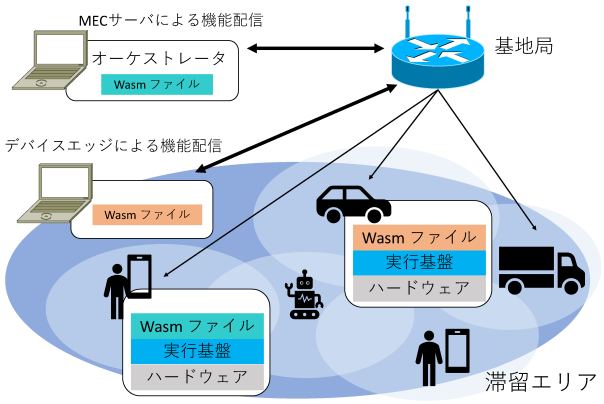


図 1　機能滞留システムの概要

3.1　機能滞留システムの目的

F-CPS では，STD とそれを処理する機能を対象のエリアに滞留させ，ユーザのニーズに応じたサービスを柔軟に提供することを目指す．先行研究である時空間データ滞留システムにより，STD の配布に関する研究が行われているが，機能の配信および実行に関してはまだ検証されていない．そこで，STD を処理するアプリケーションを滞留させることで，特定の場所，時間における柔軟なデータの利活用の促進を目的とした機能滞留システムを提案している [2] [8].

3.2　機能滞留システムの概要

図 1 に提案手法の概要を示す．まず，本システムはオーケストレータとなるサーバと，滞留エリア内のデバイスエッジ群から構成される．本システムにおける機能とは，滞留エリア内のデバイスエッジ群によって生成される STD を処理するためのアプリケーションを指す．このアプリケーションは，情報滞留システムによって滞留している STD をエッジサーバ，またはデバイスエッジ自身が分析し，モバイル端末や車両等の各ユーザの要求に応じて，複数作成することを想定している．オーケストレータは地域内で利用可能な機能を一元的に管理する役割を有しており，機能を適切な場所・時間に提供するためのスケジューリング機能を有する．

オーケストレータから各デバイスエッジへの機能の配信は，現時点で以下の 2 つのケースが想定されている．まず，一つ目に基地局に有線接続される MEC (Mobile Edge Computing) サーバにオーケストレータを設置し，収集した STD から生成した機能をエリア内に配信する場合がある．この場合，オーケストレータから基地局を介して，機能を各デバイスエッジへ配信するが，エリア内には基地局に直接繋がっていないデバイスエッジも存在すると考えられる．そこで，機能を受信できなかったデバイスエッジに対して，既に機能を所有しているデバイスエッジから情報滞留システムを用いたブロードキャスト通信による再配信を行い，機能の拡散・維持を目指す．もう一つは，エリア内のデバイスエッジが独自に機能を生成し，他のデバイスエッジに対して配信する場合がある．この場合は，基地局配下のデバイスエッジが機能の配信元となり，基地局を経由して他のデバイスエッジへと機能を配信する．このケースにおいても，基地局に接続されていないデバイスエッジが想定されるため，情報滞留システムを用いた配信も実施される．

次に，F-CPS では特定の地域内にあるデバイスエッジ上で様々な機能を実行する必要があるため，デバイスエッジ毎にハードウェアや OS に依存しないシステムが必要となる．そこで，本提案システムでは多くのデバイスが共通して有する Web 基盤上でプログラムを実行可能な Wasm を採用することで，小型で汎用性のある実行基盤を実現する．オーケストレータは自身が管理する機能を Wasm プログラムとして，ユーザの要求に応じて情報滞留システムを用いて適切な場所と時間に配布する．それを受け取ったデバイスエッジは，自身の Web 基盤上で機能を実行可能となる．これにより，地域内において STD とそれを活用する機能を滞留することが可能となり，地域に特化した F-CPS を実現可能となる．

4.　実機実験による性能評価

本研究では，Wasm を利用した機能滞留システムにおけるアプリケーション実行基盤の可搬性を評価することを目的としている．そこで，機能滞留システムにおける無線通信によってアプリケーションを転送し，受信したアプリケーションを実行するフェーズに着目し，実機による実験によって提案する実行環境の可搬性を検証する．本稿における実験では，比較手法としてコンテナ技術を活用した実行基盤を構築し，Wasm を利用した提案手法と比較することで性能を評価する．

4.1　プログラム実行環境

提案手法及び比較手法において構築したプログラム実行環境の概要を図 2 に示す．まず，提案手法である Wasm を利用した実行基盤として，Wasm 実行ファイルを Web ブラウザ外で実行可能にするランタイムとして，wasmer v3.1.1 を実装した [10] [11]. また，Wasm 実行ファイルを Open Container Initiative (OCI) イメージに変換し，OCI レジストリへと転送するツールとして，wasm-to-oci v0.1.2 を実装した [12]．比較手法であるコンテナ技術を利用した実行基盤として，コンテナ管理ツールである Docker v20.10.12 を導入し，コンテナイメージの作成及び転送，またコンテナの起動と実行の際に利用した [13]．さらに，プライベートレジストリとして，OCI ベースのイメージを保存し提供することができる harbor v2.6.1 を利用した [14] [15]．本研究では，サーバ端末を Wasm イメージ及びコンテナイメージを保存するレジストリとして使用した．

4.2　実 験 環 境

本実験では，提案手法及び比較手法の無線通信環境における可搬性を評価するため，Wi-Fi 環境と 28GHz 帯を利用するローカル 5G 環境の 2 つの実験環境で検証を行った．以下で使用機器及び 2 つの無線環境の詳細について述べる．

図 3 に実験環境を示す．本稿における評価では，3.2 節において想定される 2 番目のケースである，エリア内のデバイスエッジが他のデバイスエッジに対して機能を共有する環境を想定する．本実験では，サーバ端末とクライアント端末として，合計 4 台のノート PC (Dell latitude E5440) を利用した．また，Wi-Fi ルータとして，ASUS 社の TUF Gaming AX3000 を利用した．ローカル 5G 環境においては，ローカル 5G の基地局と

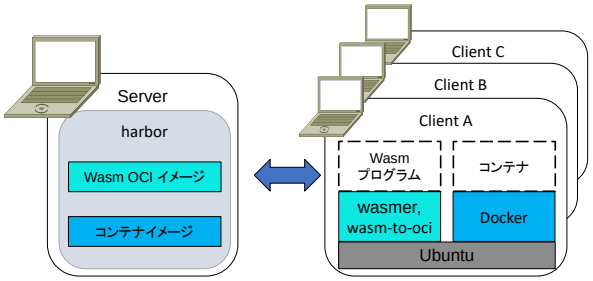


図 2　構築したプログラム実行環境

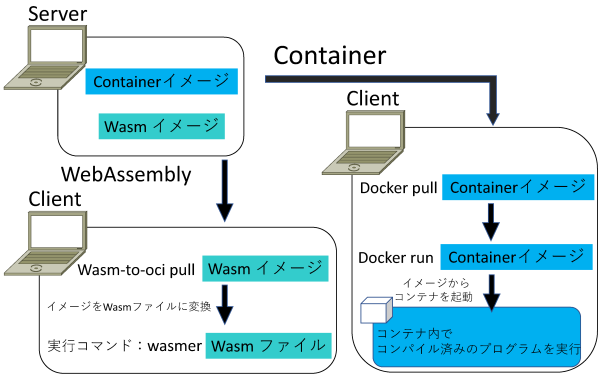
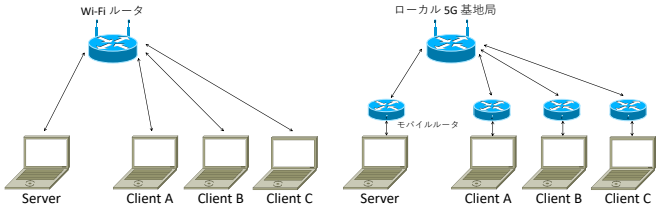


図 4　動作フロー



(a) Wi-Fi 環境　　(b) ローカル 5G 環境

図 3　実 験 環 境

無線通信を行うモバイルルーターとして，京セラ株式会社の K5G-C-100A を利用した．本稿では，サーバとデバイスエッジの両方が同一の無線通信を共有する環境において実験を行い，基地局を介した機能の配信における本提案システムの可搬性を評価する．

まず，Wi-Fi 環境では，サーバ端末 1 台とクライアント 3 台を無線通信によって接続し，通信規格として IEEE802.11ac を利用した．Wi-Fi 環境は，九州工業大学総合研究 1 号棟 8 階研究室に構築し，障害物がなく電波の干渉が少ない環境で実施した．事前実験として，iperf3 によって平均通信レートをサーバ端末と 3 台のクライアント端末でそれぞれ 10 回ずつ測定した結果，平均値は約 183Mb/s となった．Wi-Fi 環境では，Wi-Fi ルータの機能である MU-MIMO (Multi User-Multiple Input and Multiple Output) を利用している．

次に，ローカル 5G 環境について示す．ローカル 5G 環境では，サーバ端末とクライアント 3 台のそれぞれにモバイルルーターを有線接続し，DMZ 設定によって互いに 1 対 1 通信可能な環境を構築した．ローカル 5G 環境として，九州工業大学に設置された 28GHz 帯を利用する Non-Stand Alone 型のローカル 5G ネットワークを利用し，計測を実施した．こちらも事前実験として，Wi-Fi 環境と同様に iperf3 によって平均通信レートを測定した結果，平均値は約 54.3Mb/s となった．ローカル 5G 環境では，複数端末へデータ転送をする際に OFDMA (Orthogonal Frequency Division Multiple Access) が利用される．

4.3　検証方法及び動作フロー

本研究では，提案手法である Wasm による実行基盤の可搬性の評価をするため，データサイズ，データ転送時間，アプリケーション実行時間を評価指標とした．また，本実験では，F-CPS における機能を複数端末へ同時に転送した際の影響を検証するため，クライアント端末 1 台へ転送する場合と 3 台へ同時に転送する場合を計測した．

4.3.1　データサイズの検証方法

データサイズの検証として，提案手法と比較手法において，同様の処理を行うアプリケーションを作成し，そのデータサイズを測定した．提案手法では，Wasm 実行ファイルにコンパイル可能である C 言語を利用してモンテカルロ法により円周率を計算するプログラムを作成し，Wasm OCI イメージに変換して計測を行った．一方で比較手法では，提案手法と同様の C 言語による円周率計算プログラムをコンテナイメージに変換し，計測を行った．

4.3.2　データ転送時間の検証方法

データ転送時間における検証方法について記す．図 4 に本実験で実施した検証の動作フローを示す．データ転送処理は，提案手法の場合には wasm-to-oci，比較手法の場合は Docker によって実行される．データ転送の流れとして，初めにクライアント端末からサーバ端末へ pull リクエストを送信することで，受信したサーバ端末からイメージファイルが転送される．本実験では，クライアント端末で pull を実行した時間から，イメージファイルをダウンロードし，イメージファイルの展開処理が終了するまでの時間をデータ転送時間として計測を行った．複数端末へ同時に転送する際には，ntp サーバにより端末時間を同期させ，at コマンドによって同時刻に pull を実行することで検証した．1 台へ転送する場合は，各クライアント端末において 3 回ずつ計測し，3 台へ同時に転送する場合は 5 回の計測を行い，その平均値を結果とする．

4.3.3　アプリケーション実行時間の検証方法

アプリケーション実行時間は，機能として作成した円周率計算プログラムを実行し，結果を表示するまでの時間を計測した．提案手法においては Wasm 実行ファイルを受信するためコンパイル等の処理をすることなく，実行可能である．一方で，比較手法であるコンテナにおいては，コンテナイメージを起動する必要があるため，コンテナイメージを起動する時間とコンテナ内でコンパイル済みのプログラムを実行する時間の和を測定する．アプリケーションの実行は，各クライアント端末において 3 回ずつ計測し，その平均値を結果として比較する．

表 1　データサイズの比較

コンテナイメージ	Wasm イメージ
287MB	19.6kB

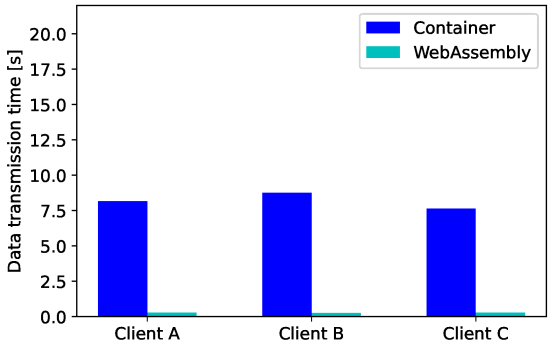


図 5　1 台ずつ (Wi-Fi 環境)

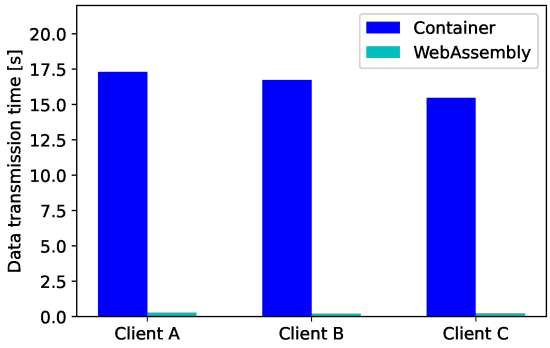


図 6　3 台同時 (Wi-Fi 環境)

4.4　実験結果及び評価

本節では，データサイズ，データ転送時間，アプリケーション実行時間のそれぞれに関する実験結果及び評価について述べる．

4.4.1　データサイズ

データサイズの結果を表 1 に示す．表 1 より，提案手法によってデータサイズを 99.993% 削減できることが明らかになった．これは，提案手法では Wasm 実行ファイルのみを OCI イメージに変換して転送できるのに対し，比較手法はイメージファイルに実行環境の設定ファイルやライブラリ等を含むためと考えられる．

4.4.2　データ転送時間

データ転送時間は Wi-Fi 環境とローカル 5G 環境の 2 つの実験環境で検証を行った．Wi-Fi 環境においてクライアント端末 1 台ずつに転送を行った場合の結果を図 5 に，3 台同時に転送した場合の結果を図 6 に示す．Wi-Fi 環境において，1 台に転送した場合のデータ転送時間の平均は Wasm が約 0.272 秒，コンテナが約 8.191 秒となり，提案手法により 7.919 秒削減された．また，3 台同時に転送した場合は Wasm が約 0.248 秒，コンテナが約 16.504 秒となり，提案手法により 16.256 秒削減された．よって，1 台の場合と 3 台同時の場合のいずれも提案手法により，データ転送時間を大幅に削減できることが明らかになった．複数端末へ転送した場合の影響において，比較手法の場合は約 2.01 倍に増加するのに対して，提案手法では約 0.912 倍となり，ほとんど変化しないことが示された．

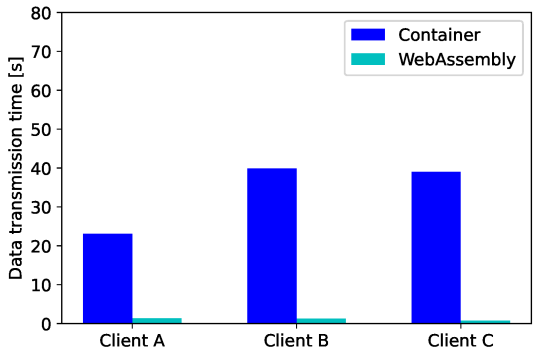


図 7　1 台ずつ (ローカル 5G 環境)

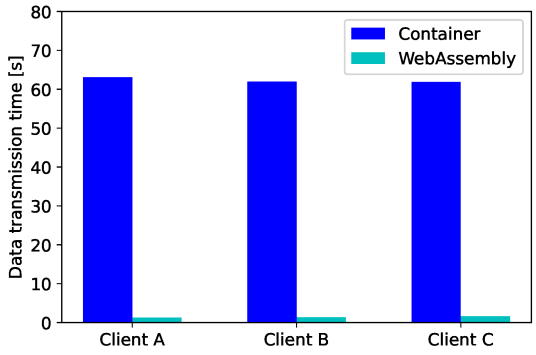


図 8　3 台同時 (ローカル 5G 環境)

ローカル 5G 環境においてクライアント端末 1 台ずつに転送を行った場合の結果を図 7 に，3 台同時に転送した場合の結果を図 8 に示す．ローカル 5G 環境において，1 台に転送した場合のデータ転送時間の平均は Wasm が約 1.170 秒，コンテナが約 24.04 秒となり，提案手法により 22.87 秒削減された．また，3 台同時に転送した場合は Wasm が約 1.445 秒，コンテナが約 62.31 秒となり，提案手法により 60.87 秒削減された．よって，Wi-Fi 環境と同様に，いずれの場合も提案手法により，データ転送時間を大幅に削減できることが明らかになった．複数端末へ転送した場合の影響は，比較手法の場合約 1.83 倍に増加したのに対して，提案手法では約 1.24 倍の増加に抑制可能であることが明らかになった．これは，Wi-Fi 環境においては MU-MIMO，ローカル 5G 環境においては OFDMA による複数端末への効率的なデータ転送が実行されたためであり，実環境における提案手法を利用した実行基盤による機能転送の可搬性が明らかになった．

4.4.3　アプリケーション実行時間

アプリケーション実行時間の結果を図 9 に示す．図 9 より，アプリケーション実行時間において，提案手法により平均して約 3.76 秒削減できることが明らかになった．これは，比較手

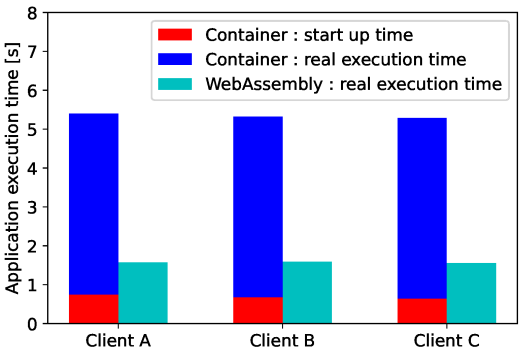


図 9　アプリケーション実行時間

法の場合，コンテナの起動処理が必要であることと，バイナリコードで記述された Wasm 実行ファイルによる数値計算処理が高速であることが理由として考えられる．よってアプリケーション実行の処理にかかる時間における提案手法の有効性が示された．

5. ま と め

本研究では，F-CPS における機能滞留を実現するため，小型で汎用性の高いアプリケーション実行基盤の構築を目的として Wasm を活用するシステムを提案した．また，Wi-Fi 環境とローカル 5G 環境の 2 つの無線通信環境において，実機によるコンテナ技術との比較検証を行い，その可搬性の評価を行った．検証実験により， Wasm を利用した実行基盤によって機能を配布した場合，データサイズを 99.993% 削減し，データ転送時間を最大 97.7% 削減することができた．また，アプリケーション実行時間においても 3.76 秒削減し，有効性を示した．これらの結果から，無線通信により複数端末に同時に機能を転送する F-CPS における提案手法の可搬性を明らかにし，Wasm による機能滞留システムが実現可能であることを示した．今後は時空間データ滞留システムとの連携や，より実用的なアプリケーションによる検証を行う．

謝辞　本研究の一部は，国立研究開発法人情報通信研究機構 (NICT) の委託研究 No.05501 による成果を含む．ここに記して謝意を表す．

文　　献

[1] Hotaka Kaneyasu, Daiki Nobayashi, Kazuya Tsukamoto, Takeshi Ikenaga, Myung Lee, “Spatio-temporal Data Retention System for Floating Cyber Physical System, ” The 10th International Symposium on Applied Engineering and Sciences (2022), Dec. 2022.

[2] 関川 柊, 佐々木 力, 野林 大起, 田上 敦士, “フローティング CPS を実現する軽量 · 高可搬な Beyond コンテナ技術の検討と課題,” 信学技報, vol. 122, no. 274, NS2022-125, 2022 年 11 月.

[3] Daiki Nobayashi, Ichiro Goto, Hiroki Teshiba, Kazuya Tsukamoto, Takeshi Ikenaga, Mario Gerla, “Adaptive Data Transmission Control for Spatio-temporal Data Retention over Crowds of Vehicles, ” IEEE Transactions on Mobile Computing, Early Access, Mar. 2021.

[4] Ichiro Goto, Daiki Nobayashi, Kazuya Tsukamoto, Takeshi Ikenaga, Myung J. Lee, “Transmission Control Method for Data Retention Taking into Account the Low Vehicle Density Environments, ” IEICE Transactions on Information Systems, Vol.E104-D, NO.4, pp.508-512, Apr. 2021.

[5] Shunpei Yamasaki, Daiki Nobayashi, Kazuya Tsukamoto, Takeshi Ikenaga, Myung J. Lee, “Efficient Data Diffusion and Elimination Control Method for Spatio-temporal Data Retention System, ” IEICE Transactions on Communications, Vol.E104-B, No.7, pp.805-816, Jul. 2021.

[6] Ichiro Goto, Daiki Nobayashi, Kazuya Tsukamoto, Takeshi Ikenaga, Myung J. Lee, “Beacon-Less Autonomous Transmission Control Method for Spatio-Temporal Data Retention, ” INCoS 2020, AISC 1263, Springer, pp.503-513, Sep. 2020.

[7] 後藤一郎ほか, “車両を用いた時空間データ滞留システムにおける送信制御手法の検討 ルクセンブルグモデル（LuST） を用いた評価 , ” 信学技報, vol.120, no.413, NS2020-132, pp.55-60, 2021 年 3 月.

[8] 関川 柊, 佐々木 力, 田上 敦士, “F-CPS に向けた高可搬型 WebAssembly コンテナの検討, ” 電子情報通信学会 2023 年 総合大会, B-6-87, 2023 年 3 月.

[9] Jonah Napieralla, “ Considering WebAssembly Containers for Edge Computing on Hardware-Constrained IoT Devices, ” Master of science Computer Science, June 2020.

[10] 2023 Wasmer Inc, “Wasmer - The Universal WebAssembly Runtime” https://wasmer.io/

[11] 2023 GitHub Inc, “wasmerio/wasmer:The leading WebAssembly Runtime supporting WASI and Emscripten” https://github.com/wasmerio/wasmer

[12] 2023 GitHub Inc, “Use OCI registries to distribute Wasm modules - GitHub” https://github.com/engineerd/wasm-to-oci

[13] 2023 Docker Inc, “Docker: Accelerated, Containerized Application Development” https://www.docker.com/

[14] Harbor Authors 2023, “Harbor” https://goharbor.io/

[15] 2023 GitHub Inc, “goharbor/harbor:An open source trusted cloud native registry project that stores, signs, and scans content. ” https://github.com/goharbor/harbor