データ収集基盤機能の動的配置手法に関する検討

羽原　拓哉*　　石井　大介　　伊藤　大輔　　緒方　祐次（日立製作所）

Dynamic Distributing Method for Data Collection Platform

Takuya Habara*, Daisule Ishii, Daisuke Ito, Yuji Ogata (Hitachi, Ltd.)

In manufacturing, it is expected that data will be shared with high reliability among multiple bases to improve operations entire the supply chain. In this research, we propose the dynamic distributing method for containerized data processing functions to collect and analyze data at multiple locations. When collecting and analyzing data from entire the supply chain, communication between the data processing function and the volume becomes a bottleneck, causing a problem of degrading the data processing. By the proposed method that dynamically determines the location of data processing functions based on the traffic volume and the communication delay time, we confirmed data processing speed is improved.

キーワード：エッジコンピューティング，コンテナ，IoT
（Edge Computing, Container, IoT）

1. はじめに

近年産業分野の現場では労働力が減少する問題に直面しており，2030 年までに 2020 年比で労働力が 10%減少すると予測されている [1]。一方，生産する製品に対するニーズが多様化しており，マスカスタマイゼーションに対応しつつ，製品の品質向上やリードタイム削減を実現することが産業分野の課題である。つまり，労働力が減少する中で生産効率を向上させることが求められている。

産業分野の現場の課題を解決する手段の一つが，様々なモノをインターネットに接続し，情報を収集・利活用する IoT 技術の活用である [2]。IoT 市場の規模は年々拡大しており，2026 年には 1 兆ドルを超過すると予測されている [3]。産業分野においても IoT 技術の応用が加速しており，生産機器の状態監視・制御，生産した製品の品質管理，トレーサビリティ，工程の最適化，作業員の作業管理など Quality, Cost, Delivery (QCD) 改善につながる様々な活用方法が検討・実装されている [4]。

製造業では，製造現場のあらゆるデータを収集し，各工程の生産性向上を図っている。例えば，製造工程にとどまらず，注文，製造，販売に至るサプライチェーン全体での業務改善を実施するという要望がある。この要望を実現するためには，サプライチェーンを構成する各拠点から発生するデータを拠点間で共有する仕組みが必要である。

本稿では，サプライチェーン上に構築する IoT 基盤が拠点間のデータ共有を実現する。図 1 に IoT 基盤を示す。IoT 基盤は IoT Gateway (GW)などを介して現場の Operational Technology (OT) データを収集・蓄積する。QCD 改善のアプリケーションの一例として，現場の状況をダッシュボード上へ可視化することが挙げられる。オペレータはダッシュボードを確認することで部品交換や工程管理など，サプライチェーン全体に渡って影響が波及する業務改善点を知ることができ，業務改善の指示をすることが可能となる。

IoT 基盤上では「可視化」や「現場へのフィードバック」など様々な IoT アプリケーションが動作する。IoT アプリケーションに応じて使用するデータ量やデータ形式が異なるため，動作するアプリケーションに応じて，収集したデータをアプリケーションが利用しやすい形式に変換したりメタデータを付与したりする必要がある。「生産機器稼働状況の可視化」や「生産順序変更の指示」などの顧客要望により使用する IoT アプリケーションが変化した場合，データ処理も変化する。顧客の要望は時々刻々と変化するため，収集されるデータの種類や量の動的な変化に対し，データの収集・分析・活用それぞれの層でデータ処理内容を柔軟に変更することが求められる。

特にデータの収集の層に関しては，メッセージングやファイル転送などのデータ収集機能を，エッジからクラウドに至るまで顧客の望む環境下で動作させなければならない。例えば，サプライチェーン上の各拠点で業務や顧客が異なるため，データ収集機能が接続する装置の要件や設置場所の制約なども異なる。IoT 基盤におけるデータ収集機能はこれらの差異を吸収する必要がある。

本稿では，IoT 基盤において，複数の拠点から発生するデータを集約するデータ収集機能に着目し，データ収集要件を達成するためのデータ収集基盤機能の動的配置手法に関する検討を行う。

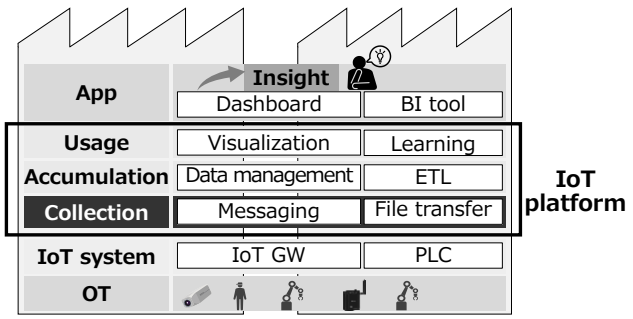

図 1　IoT 基盤

Fig. 1.　IoT platform.

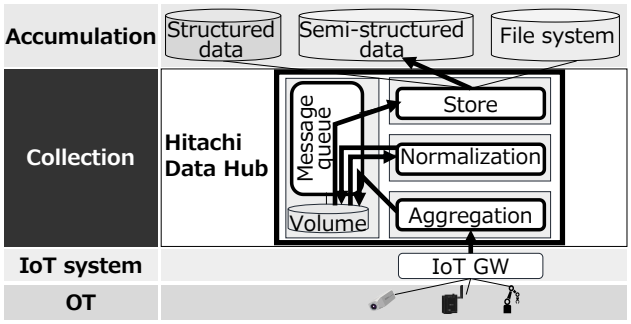

図 2　Hitachi Data Hub

Fig. 2.　Hitachi Data Hub.

2.　データ収集基盤における従来技術と課題

〈2･1〉　Hitachi Data Hub

Hitachi Data Hub は，様々な現場機器からデータを収集し，可視化・分析のためのデータ整形を実行する Publish/Subscribe メッセージング基盤である [5]。図 2 に Hitachi Data Hub のアーキテクチャを示す。Hitachi Data Hub は，データ収集機能（Aggregation）・データ加工機能（Normalization）・データ蓄積機能（Store）の 3 つの機能から構成される。データ収集機能は IoT GW などから集約するデータのデータ形式とプロトコルの差異に対応する。データ加工機能は Web Graphical User Interface ツールを用いて顧客の要望に応じたデータ加工内容を設定することが可能であり，設定内容に基づいてデータ加工を行う。データ蓄積機能は構造データ，半構造データ，ファイルシステムなど加工後のデータ形式や IoT 基盤で使用するアプリケーションに応じて柔軟に出力先を変更可能である。

Hitachi Data Hub は，データ収集機能・データ加工機能・データ蓄積機能の各機能が独立に動作する疎結合な構成を取る。各機能が独立に動作することで，顧客の要望の差異や時々刻々とした変化にデータ収集・データ加工・データ蓄積それぞれの層で個別に対応することが可能である。3 つの疎結合な機能を接続する役割を果たすのがメッセージキューである。メッセージキューは取り扱うデータをボリュームに格納するため，メッセージキューに入出力がある度にメッセージキューはボリュームへデータを読み書きする。

図 2 中の矢印は IoT GW からデータを送信しデータが蓄積されるまでの流れを示す。データ収集機能・データ加工機能・データ蓄積機能の 3 つの機能を順に使用するが，各機能を使用する度にメッセージキュー及びボリュームを経由する。従って，メッセージキューとボリューム間のトランザクションが Hitachi Data Hub 全体のデータ処理能力に影響を及ぼす。

〈2･2〉　コンテナアーキテクチャの概要

コンテナは，実行環境の Operating System (OS) やインフラ基盤に依らずアプリケーションを稼働可能とするアプリケーションの仮想化技術の一つである。コンテナを使用する際にはホストマシンのハードウェア，OS の上にコンテナ管理ソフトウェアであるコンテナエンジンを配置する。

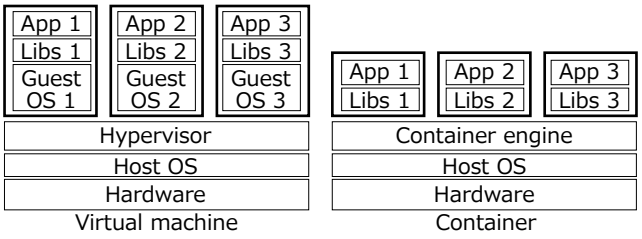

図 3　仮想化技術の比較

Fig. 3.　A comparison of virtualization technology.

図 3 に仮想マシンとコンテナの比較図を示す。仮想マシンでは OS を含めた仮想化が行われるため，仮想マシン内に Guest OS が存在する。OS やハードウェアリソースを個別に利用することができる利点があるが，Guest OS を含むため，仮想マシンのイメージサイズは大きくなり，一般的にギガバイトオーダーとなる。また，仮想マシンを起動する際には Guest OS の起動が必要なため，分オーダーの時間を要する。

一方コンテナはコンテナ内に疑似的に OS を持つが，実態はホストマシンの OS を使用する。コンテナの場合，起動・終了はホストマシン視点では一つのプロセスの動作に過ぎない。従ってコンテナは一般的に秒オーダーと短時間で起動可能である。また，コンテナイメージに OS を含まないため，コンテナのイメージサイズは仮想マシンと比較して小さく，一般的にメガバイトオーダーとなる。

短時間での起動が可能でありイメージサイズが小さいというコンテナの特徴を活用することで，可用性を確保できる。例えばアプリケーションのアップデートやトラブルによる停止が発生する際には，同一のコンテナを正常に動作するマシン上に素早く立ち上げることでユーザ視点ではアプリケーションが停止しない。この例のように，コンテナは起動後に同じマシンで稼働し続けず，停止・更新・再配置される前提で運用される。

コンテナの停止・更新・再配置に備えるため，コンテナはステートレスであることが求められる。ステートフルなアプリケーションではステートをコンテナ外に複製して保持する必要がある [6]。データ収集基盤である Hitachi Data Hub においては収集・加工・蓄積したデータをコンテナ外に保持することでステートレスなコンテナとなる。

図 4 にコンテナでのデータ保持方式を示す。10 秒ごとにデータを収集し，Data4 を収集した直後に App1 をアップデートした例を示す。コンテナ内にデータを格納する図 4 左の方式では，App1 のアップデートに伴うコンテナの停止・再配置により，App1 をアップデートするまでに収集した Data1 から Data4 までのデータが消失し，App1 のアップデート後は Data5 以降のみが利用可能となる。

図 4 右の方式では，コンテナの停止・更新・再配置によりコンテナ内のデータが消失しないようにするため，Persistent Volume と呼ばれるコンテナ外部のデータ格納領域を利用する。Persistent Volume を使用する図 4 右の方式では，App1 をアップデートするまでに収集した Data1 から Data4 までのデータが Persistent Volume 上に格納されており，Data1 以降のすべてのデータを再配置後のコンテナから利用可能であることを例示している。Persistent Volume を使用することでコンテナ自体がステートレスとなり，コンテナの停止・更新・再配置によるデータ消失が発生しない。

〈2･3〉 アプリケーションデプロイ基盤

様々な顧客に IoT 基盤を活用したソリューションを提供するために，ソリューション展開が容易なデプロイ基盤が必要である。デプロイ基盤において業界標準のコンテナ技術である Docker やコンテナオーケストレーションツールである Kubernetes を採用することで，様々な IT インフラの環境の差異に依らずコンテナを起動することが可能となり，アプリケーションの構築が容易となる [7]。

〈2･4〉 拠点間のデータ共有を伴うデータ処理における課題

1 章で述べた，サプライチェーン全体での業務改善を実現するという要件に対し，従来技術を用いてサプライチェーン全体を横断してデータ収集を行う IoT 基盤を構築する。このとき，データ処理を行うコンテナと Persistent Volume 間のデータ通信がデータ処理のボトルネックとなり，データ処理速度の低下につながることが課題であることを見出した。以下，図 5 を用いて課題の詳細を示す。

図 5 にデータ収集事例の一つとして，Hitachi Data Hub を用いて親工場，子工場の 2 つの工場間でデータ共有を行う例を示す。この実施例では各機能をコンテナ化した Hitachi Data Hub を使用する。IoT アプリケーションの用途が「サプライチェーン全体での最適化」である場合，工場間でデータを共有する必要が生じる。子工場での製造結果に関するデータが Hitachi Data Hub により収集され，親工場に共有される。親工場では親工場での製造結果に関するデータが Hitachi Data Hub により収集される。子工場と親工場の製造結果に関するデータを用いて分析が行われ，拠点を跨いだ指示が生成される。図 5 では親工場の生産速度と比較し子工場の生産速度が速いため，「子工場の生産スピードを落とす」指示が子工場に送信される例を示す。

拠点を跨いでデータ共有を行う際，データ処理を行うコンテナとデータを格納する Persistent Volume が異なる拠点に配置される可能性があり，コンテナと Persistent Volume 間のデータ通信がデータ処理のボトルネックとなる。製造現場においてサプライチェーン全体での業務改善を実現するためには，0.6 秒ごとにデータが発生するデバイス 60 台からすべてのデータを収集するというデータ収集要件が存在する。即ち，1 秒当たり 100 データのデータ処理を行うことが求められる。

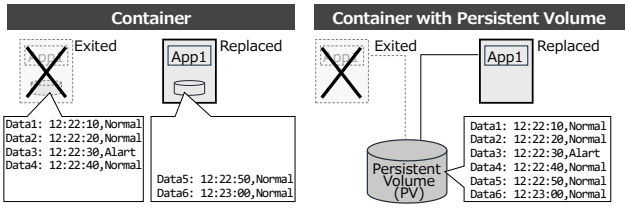

図 4　コンテナでのデータ保持

Fig. 4.　Data preservation with container.

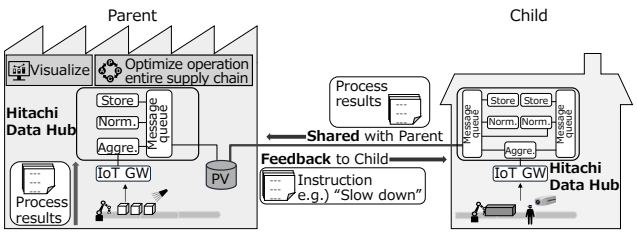

図 5　拠点間のデータ共有

Fig. 5.　Data sharing between sites.

そこで本稿では Hitachi Data Hub を複数工場に導入し工場間でのデータ共有を行う状況下で，データ処理のボトルネックの解消を実現するコンテナ配置手法を提案する。更に，提案手法が 1 秒当たり 100 データのデータ処理を実現可能か評価する。

3. エッジオーケストレータを用いた動的コンテナ配置手法

〈3･1〉 エッジオーケストレータの概要

コンテナの配置場所を決定する際にエッジオーケストレータを用いる。図 6 にエッジオーケストレータの概要を示す。エッジオーケストレータは顧客要望やトラフィック情報を元にコンテナの新規配置計画を作成し，コンテナの再配置を実行する。デバイスの追加や利用するアプリケーションの変更などの顧客要望は，エッジオーケストレータがユーザの入力により取得する。通信量や通信遅延時間などのトラフィック情報は，エッジオーケストレータがネットワークを測定して取得する。

以下，データ集約の制約とトラフィック情報を用いたトラフィック評価方法について説明する。

〈3･2〉 データ集約の制約

図 7 に本稿におけるデータ集約の制約を示す。Hitachi Data Hub の各機能はコンテナ化され，親工場，子工場に配置されている。IoT GW は各工場の機器からデータを収集し，Hitachi Data Hub のデータ収集機能に転送する。本稿

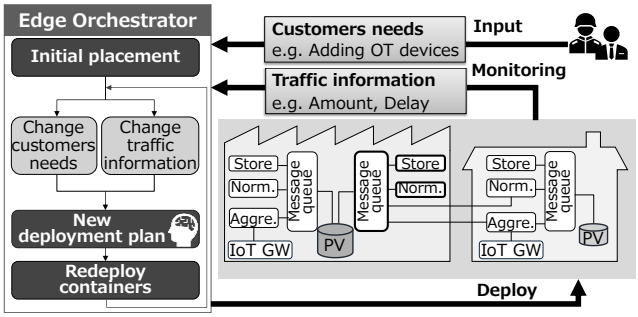

図 6　エッジオーケストレータの概要

Fig. 6.　Overview of Edge Orchestrator.

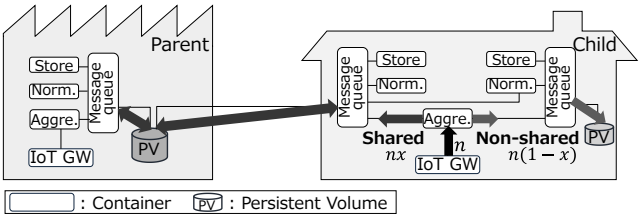

図 7　データ集約の制約

Fig. 7.　Data aggregation constraint.

では，収集するデータを共有データと非共有データに分類する。共有データはサプライチェーン全体の改善のために使用されるデータであり，Hitachi Data Hub は共有データを親工場の Persistent Volume へ格納する。一方非共有データは各工場の改善のために使用されるデータであり，Hitachi Data Hub はそれぞれの子工場の Persistent Volume に格納する。Hitachi Data Hub の機能の内，データ加工機能，データ蓄積機能の 2 機能とメッセージキューは共有データと非共有データを分離して扱うため 2 コンテナずつ稼働する。図 7 中のnは IoT GW から送信される 1 秒当たりデータ数を，xは共有データの割合を示し，$0 \leq x \leq 1$とする。共有データの割合xは顧客の要望により変動するパラメータである。

〈3･3〉　トラフィック評価方法

図 8 に本稿における Hitachi Data Hub を使用したデータ収集・分析モデルを示す。3.1 節で説明したコンテナの配置先を決定するエッジオーケストレータは，コンテナの新規配置計画を作成するため，コンテナ間，コンテナと Persistent Volume 間のリンクに名前を付け，各リンクに対してトラフィック情報である通信量と遅延時間を測定する。表 1 に図 8 で示したモデルにおける Link1 から Link10 の通信量と遅延時間を示す。なお，$n = 600$，$x = 0.8$である。

Hitachi Data Hub は IoT GW から入力されたデータに対し，データ収集機能・データ加工機能・データ蓄積機能を用いてデータ処理を行う。共有データを扱う場合には親工場からのフィードバックを受信するため，図 8 の丸囲い 4 番で示した非共有データ用のデータ加工機能も用いる。非共有データ用のデータ加工機能は共有データ用の Persistent Volume からデータを取得し，データ処理を行い，非共有データ用の Persistent Volume に格納するデータ処理を実行する。

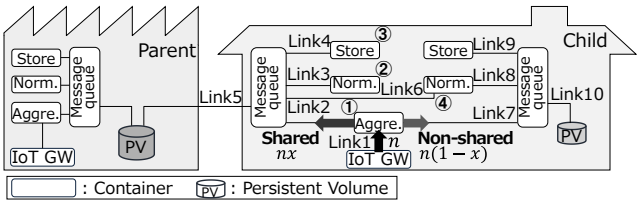

図 8　データ収集・分析モデル

Fig. 8.　Data collection and analysis model.

表 1　トラフィック情報

Table 1.　Traffic information.

Link name	Traffic amount (byte)	Traffic delay (msec)
Link1	$\chi_{L1} = 600$	$\delta_{L1} = 0.4$
Link2	$\chi_{L2} = 480$	$\delta_{L2} = 0.4$
Link3	$\chi_{L3} = 1480$	$\delta_{L3} = 0.4$
Link4	$\chi_{L4} = 1000$	$\delta_{L4} = 0.4$
Link5	$\chi_{L5} = 3960$	$\delta_{L5} = 6$
Link6	$\chi_{L6} = 1000$	$\delta_{L6} = 0.4$
Link7	$\chi_{L7} = 120$	$\delta_{L7} = 0.4$
Link8	$\chi_{L8} = 1330$	$\delta_{L8} = 0.4$
Link9	$\chi_{L9} = 250$	$\delta_{L9} = 0.4$
Link10	$\chi_{L10} = 1700$	$\delta_{L10} = 0.4$

Hitachi Data Hub の各機能はメッセージキューとメッセージキューのデータを格納する Persistent Volume を介してデータ処理を行う。例えば図 8 の丸囲い 2 番のデータ加工機能は「親工場の Persistent Volume からデータを取得」「データ加工」「親工場の Persistent Volume へ処理後のデータを送信」の 3 段階でデータ処理を行う。このとき Link3 を 3 回，Link5 を 3 回使用する。従ってデータ加工機能の処理遅延時間t_{norm}は

$$t_{\mathrm{norm}}=3\delta_{L3} + 3\delta_{L5} \quad (1)$$

と求められる。同様にデータ収集機能の処理遅延時間t_{aggre}，データ蓄積機能の処理遅延時間t_{store}，フィードバック用データ加工機能の処理遅延時間$t_{\mathrm{norm_FB}}$は

$$t_{\mathrm{aggre}}=\delta_{L1} + \delta_{L2} + \delta_{L5} \quad (2)$$

$$t_{\mathrm{store}}=2\delta_{L4} + 2\delta_{L5} \quad (3)$$

$$t_{\mathrm{norm\_FB}}=2\delta_{L5} + 2\delta_{L6} + \delta_{L8} + \delta_{L10} \quad (4)$$

と求められる。

続いて，データ処理のボトルネックとなる機能の特定を行う。1 つの共有データを処理可能な時間間隔τ_1は

$$\tau_1 = \max\{t_{\mathrm{aggre}}, t_{\mathrm{norm}}, t_{\mathrm{store}}, t_{\mathrm{norm\_FB}}\} \quad (5)$$

と求められる。本例では表 1 に示す遅延時間と式(1)から(5)を用いて

$$\tau_1 = t_{\mathrm{norm}} = 19.2 \quad (6)$$

と求められる。従って，共有データ用のデータ加工機能がデータ処理のボトルネックとなる機能と特定される。非共有データも同様に算出する。非共有データ用のデータ収集機

能の処理遅延時間$t_{\mathrm{aggre}'}$，非共有データ用のデータ加工機能の処理遅延時間$t_{\mathrm{norm}'}$，非共有データ用のデータ蓄積機能の処理遅延時間$t_{\mathrm{store}'}$はそれぞれ

$$t_{\mathrm{aggre}'}=\delta_{\mathrm{L1}}+\delta_{\mathrm{L7}}+\delta_{\mathrm{L10}} \tag{7}$$

$$t_{\mathrm{norm}'}=3\delta_{\mathrm{L8}}+3\delta_{\mathrm{L10}} \tag{8}$$

$$t_{\mathrm{store}'}=2\delta_{\mathrm{L9}}+2\delta_{\mathrm{L10}} \tag{9}$$

と求められる。1 つの非共有データを処理可能な時間間隔τ_2は非共有データ用のデータ収集機能の処理遅延時間$t_{\mathrm{aggre}'}$，非共有データ用のデータ加工機能の処理遅延時間$t_{\mathrm{norm}'}$，非共有データ用のデータ蓄積機能の処理遅延時間$t_{\mathrm{store}'}$を用いて

$$\tau_2=\max\{t_{\mathrm{aggre}'},t_{\mathrm{norm}'},t_{\mathrm{store}'}\} \tag{10}$$

と求められる。

最後にデータ処理能力を算出する。共有データの発生頻度z_1と非共有データの発生頻度z_2は 1 秒当たりデータ発生数nと共有データ割合xを用いて求められる。

$$z_1=\frac{1}{nx} \tag{11}$$

$$z_2=\frac{1}{n(1-x)} \tag{12}$$

共有データと非共有データを合わせて 1 秒当たり処理可能データ数Nは

$$N=n\cdot\min\left\{\frac{z_1}{\tau_1},\frac{z_2}{\tau_2}\right\} \tag{13}$$

と求められる。

エッジオーケストレータはボトルネックを解消するためにコンテナの配置場所を変更する。このとき，通信量と遅延時間を元に計算されるスコアを用いてコンテナの配置場所を決定する。各リンクのスコア$r_{\mathrm{L}k}$と，システム全体のスコアRを以下のように定義する。

$$r_{\mathrm{L}k}=\chi_{\mathrm{L}k}\cdot\delta_{\mathrm{L}k} \tag{14}$$

$$R=\sum_{k=1}^{m}r_{\mathrm{L}k} \tag{15}$$

ただし，mはエッジオーケストレータが管理するシステム内のリンク数を表す自然数である。エッジオーケストレータはシステム全体のスコアRが最小となるようにコンテナの新規配置計画を作成し，コンテナの再配置を実行する。

図 9 にコンテナ再配置を実行後のコンテナの配置場所を示す。表 2 にコンテナ再配置前後の各機能の処理遅延時間を示す。コンテナ再配置前にボトルネックであった共有データ用データ加工機能の処理遅延時間が 19.2 msec から 2.4 msec へ短縮され，ボトルネックが解消された。一方，非共有データ用のデータ加工機能の処理遅延時間が 2.4 msec から 19.2 msec へ拡大した。本例では$x=0.8$であり，非共有データの割合が小さく，非共有データ用のデータ加工機能の処理遅延時間が拡大してもデータ処理速度の影響を及ぼさない。1 秒当たり処理可能データ数Nはコンテナの再配置により 65.1 から 156.3 となり約 2.4 倍となった。

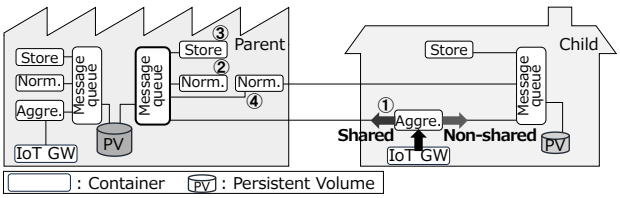

図 9　コンテナ再配置結果

Fig. 9.　Container relocation result.

表 2　処理遅延時間

Table 2.　Processing delay time.

Function	Processing delay time (msec)	
	Before redeployment	After redeployment
Aggregation for Shared	t_{aggre} = 6.8	t_{aggre} = 6.8
Normalization for Shared	t_{norm} = 19.2	t_{norm} = 2.4
Store for Shared	t_{store} = 12.8	t_{store} = 1.6
Normalization for feedback	$t_{\mathrm{norm\_FB}}$ = 13.6	$t_{\mathrm{norm\_FB}}$ = 8
Aggregation for Non-shared	$t_{\mathrm{aggre}'}$ = 1.2	$t_{\mathrm{aggre}'}$ = 1.2
Normalization for Non-shared	$t_{\mathrm{norm}'}$ = 2.4	$t_{\mathrm{norm}'}$ = 19.2
Store for Non-shared	$t_{\mathrm{store}'}$ = 1.6	$t_{\mathrm{store}'}$ = 1.6

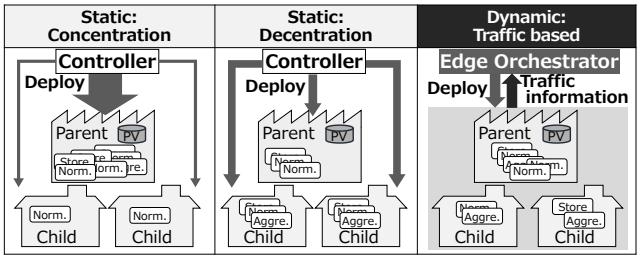

図 10　コンテナ配置モデル

Fig. 10.　Container deployment model.

4.　評価

〈4･1〉　評価モデル

3 章で述べた動的コンテナ配置手法を評価する。図 10 に評価に用いるコンテナ配置モデルを示す。静的集約モデル (Static: Concentration) では Hitachi Data Hub のコンテナの内，非共有データ用のデータ加工機能，データ蓄積機能，メッセージキューのコンテナを子工場に，それ以外のコンテナを親工場へ集約して配置する。親工場と子工場の規模に大きな差があり，親工場に多くの計算機リソースがあるケースを想定している。静的非集約モデル(Static: Decentration)では各拠点に共有データ用のコンテナ及び非共有データ用のコンテナを配置する。親工場と子工場の規模に大きな差がなく，同程度の計算機リソースがあるケースを想定している。図 8 がこのモデルに該当する。動的トラフィックベースモデル (Dynamic: Traffic based) は 3 章で述べたエッジオーケストレータを用いたコンテナ配置手法である。トラフィック情報を元にスコアを算出し，システム全体のスコアRが最小となるようにコンテナを配置する。

表 3 にデータ収集要件を元に設定したシミュレーション評価諸元を示す。0.6 秒ごとにデータが発生するデバイス 60 台からすべてのデータを収集することがデータ収集要件で

表 3 シミュレーション評価諸元

Table 3. Simulation specifications.

Parameter	Value
Number of factories	10
Devices at each site	60
Desired data generation interval	0.6 sec
Traffic delay between sites	6 msec
Traffic delay within the site	0.4 msec

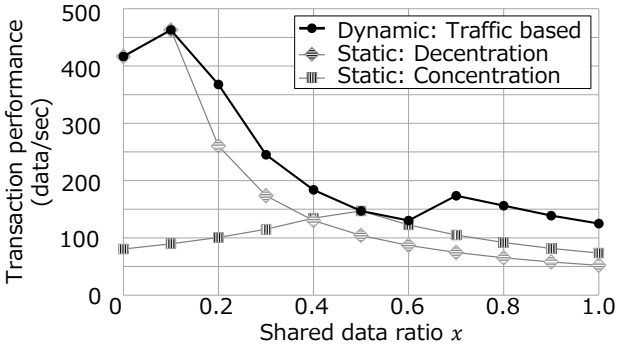

図 11 トランザクション処理能力

Fig. 11. Transaction performance.

ある。即ち，1 秒当たり 100 データの収集を行うことが必要である。

〈4･2〉 シミュレーション評価

図 11 にシミュレーション評価結果を示す。横軸は共有データの割合xを表し，縦軸は 1 秒当たり処理可能データ数Nを示す。4.1 節で述べたデータ収集要件に照らすと，1 秒当たり処理可能データ数Nの所望値は 100 である。

静的集約モデルと静的非集約モデルは，共有データの割合xによっては所望値である 100 を下回った。所望値を下回ることは現場から収集できないデータが存在することを意味する。一方，動的トラフィックベースモデルでは，共有データの割合xに依らず常に所望値である 100 を上回った。通信量と遅延時間を元にコンテナを配置する提案手法を用いることで，複数拠点間でのデータ共有を実現しつつ所望のデータ処理速度を達成した。本手法を用いることで，サプライチェーン全体での業務改善を実施するというデータ収集要件に応える見込みを得た。

〈4･3〉 実機検証

Kubernetes 上で動作する Hitachi Data Hub のコンテナ間の通信に疑似的に遅延を発生させ，複数拠点間のデータ収集を模擬したデータ収集の実機検証を行った。同一拠点内通信の遅延時間は 0ms，拠点間通信の遅延時間は 20ms に設定した。$x = 0.8$において，静的非集約モデルと動的トラフィックベースモデルを適用したときのコンテナ配置は，シミュレーション評価によるとそれぞれ図 8，図 9 となる。

表 4 に各コンテナ配置モデルを適用した際の 1 秒当たり処理可能データ数Nを示す。動的トラフィックベースモデルを適用したときは静的非集約モデルを適用したときと比較し，1 秒当たり処理可能データ数が 41.8%増加した。計算機リソースを増強し，Hitachi Data Hub の各機能コンテナを複数並列処理することでも 1 秒当たり処理可能データ数を向上させることができる。しかし，本評価により，同じ計算機リソースであっても動的トラフィックベースモデルを適用して適切なコンテナ配置を行うことで，1 秒当たり処理可能データ数を向上させることができる見込みを得た。

表 4 実機検証結果

Table 4. Actual machine verification result.

Container deployment model	Transaction Performance (data/sec)
Static: Concentration (Fig. 8)	11.7
Dynamic: Traffic based (Fig. 9)	16.6

5. おわりに

本稿では複数拠点間でデータ共有を行う IoT 基盤におけるデータ処理機能のコンテナ配置手法について検討を行った。IoT 基盤におけるデータ収集機能として，コンテナ化された Hitachi Data Hub を対象として検討を行った。コンテナ化された Hitachi Data Hub の各機能と，Hitachi Data Hub が取り扱うデータが格納される Persistent Volume との間の通信がボトルネックとなり，データ処理能力が低下することに着目した。データ処理能力低下を防ぐため，トラフィック情報を指標にデータ処理機能の配置場所を動的に決定する手法を提案した。

シミュレーション評価により，提案手法を用いることでデータ共有度の変化に依らずデータ収集要件であるデータ処理速度 100 データ毎秒を達成し，複数拠点間でのデータ共有を実現可能であることを明らかにした。また，実機検証により提案手法が 1 秒当たり処理可能データ数向上に有効であることを明らかにした。

顧客要望に柔軟に対応する IoT 基盤を構築するためには，計算機リソースやトラフィック量の多寡や変動など，現場の状況にリアルタイムに反応することが求められる。現場の状況に応じて適切なコンテナ配置を行う手法を確立することが今後の課題である。

文　献

（1） 独立行政法人 労働政策研究･研修機構 ：「労働力需給の推計―労働力需給モデル（2018 年度版）による将来推計―」（2019）

（2） M. Chiang, T. Zhang, “Fog and IoT: An Overview of Research Opportunities,” IEEE Internet of Things Journal, Vol. 3, No. 6 (2016)

（3） Fortune Business Insights, “Internet of Things (IoT) Market Size,” https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307

（4） B. Chen, et al., “Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges,” IEEE Access, Vol. 6 (2017)

（5） 伊藤大輔，石田仁志 ：「生産現場を改善し続けるデジタルツイン技術」，日立評論，Vol. 102，No. 3，pp.358-362 (2020)

（6） L. A. Vayghan, et al., “Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes,” IEEE 19th International Conference on Software Quality, Reliability and Security (2019)

（7） 羽原拓哉，石井大介，緒方祐次 ：「IoT データ収集処理機能の動的機能配備を実現するコンテナインテグレーションに関する検討」，電気学会通信研究会，CMN-19-020，pp.29-34(2019)