

Fiware を用いた都市 OS のためのデータブローカー仮想化手法

鄭在勝¹ 郡浦宏明¹ 栖川淳¹ 木下雅文¹

概要: 様々なデータを活用して都市の課題を解決するスマートシティでは、データ連携を容易にするための基盤である都市 OS が必要とされている。都市 OS 開発工数を削減する OSS(Open Source Software)として欧州で実績のある Fiware が期待されているが、そのメッセージ仲介を担う OCB(Orion Context Broker)は、人口 300 万人規模の都市で想定した要件 (250 msg/s) を満たすために、機能を維持しながら、スループットをスケールさせることが課題である。そこで、本研究では、データ領域を分け持った複数の OCB を一つに扱い、スループットをスケールさせる仮想化手法を開発した。評価の結果、要件を満たしスループットが 1300 msg/s までスケールすることを確認した。

Data Broker Function Virtualization Method for Scalable City OS Based on Fiware

JAESEUNG JEONG¹ HIROAKI KONOURA¹ JUN SUGAWA¹
MASAFUMI KINOSITA¹

1. はじめに

2016 年に決定された第 5 期科学技術基本法では、ネットワーク化やサイバー空間利用の飛躍的発展といった潮流を踏まえ、サイバー空間の積極的な利活用を中心とした取組を通して、新しい価値やサービスが次々と創出され、社会の主体たる人々に豊かさをもたらす超スマート社会を未来社会の姿として共有し、その実現に向けた一連の取組を Society 5.0 として推進し、世界に先駆けて超スマート社会を実現していくことを示した[1]。スマートシティは、Society 5.0 を社会的に実装する場とされている[2]。国土交通省は、スマートシティを、都市の抱える諸課題に対して、ICT 等の新技術を活用しつつ、マネジメント(計画、整備、管理・運営等)が行われ、全体最適化が図られる持続可能な都市または地区と定義している[3]。2010 年頃のスマートシティは、エネルギーを始めとした個別分野特化型の取り組みが中心だったが、近年では、エネルギー・交通・医療などの多種多様なデータを活用し、複数分野に幅広く取り組む分野横断型の取り組みが増加している[3]。

スマートシティを実現するサービスのための共通機能が集約され、様々なデータを分野横断的に収集・連携、サービス導入・連携を容易にするための基盤として、都市 OS が必要とされている[2][4]。都市 OS は共通 API 等の体系的な相互運用機能を持っており、システム面でスマートシティを効率的に低コストで実現することができる。また、各都市でデータやサービス連携のための共通機能を持っており、複数の都市や分野が連携するサービスの開発を行うことが可能となる[5]。

日本経済団体連合会は、都市・他分野のデータプラットフォームとの相互連携や横展開を容易にするために、実績のある OSS(Open Source Software)の採用が不可欠であると述べている[6]。都市 OS の構成としても、OSS は必要であり、日本経済団体連合会では、その例として Fiware を挙げている。EU の官民連携プロジェクトで開発されたプラットフォームである Fiware では、役割が異なる複数の GEs(General Enablers)と呼ばれているコンポーネント群があり、それらを必要に応じて組み合わせることでシステムを開発できる[7]。GE 間のデータやり取りは、標準化されたオープン API である NGSI (Next Generation Service Interface) を用いて行われる。2018 年の時点で、世界中で 115 個の都市で Fiware を採択しており、日本でも Fiware を用いたデータ連携基盤を開発している[8]。

本研究では、スマートシティの代表的なユースケースを挙げ、Fiware ベースの都市 OS において、Fiware のメッセージブローカー(OCB)の性能問題を明らかにし、OCB の機能を維持しながら性能向上させるための、メッセージブローカー機能の仮想化技術を検討し、評価した。

2. 都市 OS のデータブローカー

2.1 都市 OS のメッセージブローカー機能の概要

本節では、都市 OS で必要とされている共通機能に関して紹介し、本研究で注目しているデータ仲介機能(ブローカー)について述べる。内閣府ではスマートシティリファレンスアーキテクチャを公開しており、都市 OS に必要な共通機能を述べている[5]。スマートシティリファレンスアーキテクチャで述べているデータ仲介は、メッセージブローカ

¹ 株式会社 日立製作所 研究開発グループ

一の機能であり、データ提供者（デバイス）やデータ利用者（サービス）に対して、データ登録・参照・更新・削除ができるようにする機能である。メッセージブローカーにより、サービスが都市 OS のデータを利用することができる。

2.2 都市 OS のメッセージブローカー機能の要件

本節では、スマートシティのユースケースを挙げ、都市 OS のメッセージブローカーの要件について述べる。スマートシティは、都市の課題を解決し、市民の QoL(Quality Of Life)を向上させるための様々なサービスを提供している。McKinsey & Company は、都市の課題として、環境問題や混雑回避、病院負荷低減などを挙げており、これらを解決するスマートシティの取り組みが必要だと述べている[9]。また、スペインのバルセロナ市は環境問題やエネルギーなどのインフラの効率化を課題としてスマートシティを実現しており[10]、サンタンデル市でも、環境問題や交通負荷低減に取り組んでいる[11]。以上を踏まえ、本研究では、環境問題や交通・病院負荷の低減のためのサービスであり、様々なデータや多数のデバイス数を必要とされる、“コロナ感染予防マップ”、“エネルギーマネージメント”、“環境モニタリング”といった仮想のサービスを挙げる。以下で述べるサービスの機能やデバイス設置場所は想定のもとで述べており、サービス要件は、日本の人口 300 万人規模の都市に開始する場合、必要なデバイス数やスループットなどを想定して定めた。本節で定めるデバイス数は、人口 300 万人規模の都市の施設数と設置される個所の数を想定して計算したものである（例：500(駅数) × 4(改札, ホームなど設置区間数) = 2000)。スループットに関しては、1 デバイスから 1 データが送信される前提で、デバイス数と更新頻度から計算したものである。

「コロナ感染予防マップ」は、駅、官公庁、公園、体育館、大通りなどに設置されたカメラセンサから計測された混雑度データを用いて、混雑状況の可視化や、混雑している場所の通知をするサービスである。混雑度データは時間の変化に大きく影響するので、サービスの質を向上させるためには、頻繁に更新する必要がある、更新頻度を 1 分と想定した。デバイス数に関しては、駅(500(駅数) × 4(改札, ホームなど設置区間数) = 2000)、官公庁・公共施設 (500(施設数) × 2(区間数) = 1000)、大通り(1000(大通り数) × 2(区間数) = 2000)を合計して 6000 個とした。デバイス数と更新頻度を考慮して、必要なスループットは 100 msg/s となる。また、混雑している場所を通知するためには、データをサービスにすぐ渡さないといけないので、常にデータを取得できるようにする常時監視が必要である。

「エネルギーマネージメント」は駅、官公庁、公園、体育館などのスマートメータから収集された電力使用量データを用いて、施設の電力使用量を可視化するサービスである。電力使用量を統計して可視化すると予想され、データを頻繁に取得する必要はない。したがって、30 分周期でデ

ータが更新されるとした。デバイス数に関しては、駅 (500(駅数))、官公庁・公共施設 (500(施設数))を合計して 1000 個とした。これらを考慮して、要求スループットを 0.5 msg/s としている。一定時期でデータの統計をとるので、一定時間ごとにデータを取得する定期監視すると考えられる。

「環境モニタリング」は、街灯に設置されたセンサから計測された CO2 濃度や騒音データを用いて、都市中の CO2 濃度や騒音程度の可視化や、任意の閾値を超えたときの通知をするサービスである。都市の環境を常に監視するために、1 分の更新頻度を想定される。デバイス数は、街灯が 30 m 間隔で 30000 個程度あり、約 100m ごとに計測する想定で、10000 個とした。更新頻度とデバイス数から必要スループットは 150 msg/s である。また、通知するためには、データの値を常に監視する必要があるため、常時監視が必要である。

挙げている 3 つのサービスを踏まえると、都市 OS のメッセージブローカーに要求されるスループットは、250 msg/s である。ここで、これからサービスの数や、デバイスの数が増えることが予想されており、第 1 要件として、スループットは 250 msg/s を超えてスケールする必要がある。また、挙げている 3 つのサービスは、データを取得する方法が異なっており、第 2 要件として、メッセージブローカーのデータ提供方法は常時監視と定期監視に対応できる必要がある。

2.3 Fiware の Orion Context Broker を用いたデータ連携

本節では、Fiware のメッセージブローカー機能である Orion Context broker(OCB)のデータ連携のための機能を述べる。Fiware を構成しているコンポーネントを GE と呼んでおり、30 種類の GE の中で核となるのが OCB である。Fiware を用いたシステムは、OCB を中心で必要に応じて他の GE を OCB につなげて開発される。OCB はデータの登録・参照・更新・削除や、サブスクリプションと通知の機能が可能である。サービスはデータを取得するときに、データを参照する (Pull) か、必要データをサブスクリプションして通知で受信する、2 つの方法がある (図 1)。Fiware は、OCB が処理するデータやサブスクリプション情報を保存するためのデータベースとして、MongoDB を採用している。OCB でデータを扱うときは、まず MongoDB にデータフィールドを作成するために、データを登録する作業が必

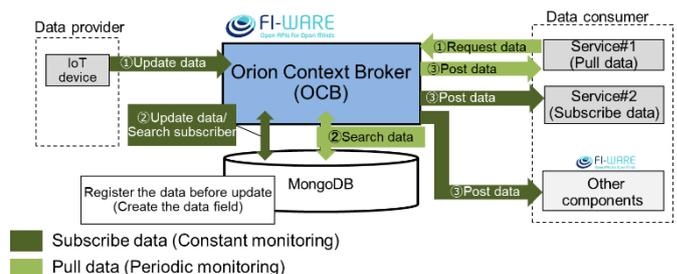


図 1 OCB のデータ登録・参照・更新・サブスクリプション

要である。登録されているデータに対して、データ提供者がデータを更新すると、OCBはMongoDBにデータを更新した後に、そのデータをサブスクリプションしているデータ利用者に通知する。データが更新されたときに、サービスは通知ですぐデータを取得することができるので、サブスクリプション・通知機能は常時監視に対応できる。登録されているデータに対して、サービスがデータ参照リクエストをすると、OCBは参照要求されたデータをMongoDBから検索し、そのデータをサービス側に送信できる。この機能を用いて、サービスが定期的にデータ要求を投げることで、定期的にデータを取得することができるので、定期監視に対応できる。

上記を踏まえると、OCBの機能を活用することで、2.2節で述べた都市OSのメッセージブローカーの要件の中で、データ提供方法は常時監視や定期監視に対応する必要があることを満たすことができる。

2.4 Orion Context Broker を用いた都市 OS の課題

本節では、OCBが2.2節で述べた都市OSメッセージブローカーのスループット要件を満たしているかを確認し、OCBを用いた都市OSの課題について述べる。2.2節で述べたサービスの中で、「エネルギー管理」のみデータ参照(Pull)を用いてデータを取得し、その頻度も高くないので、性能評価のシナリオとして、Pullは行わずに、データ更新のみにした。

MongoDBに登録されたデータ数がOCBの性能に与える影響を確認した。2core,8GBメモリのサーバ複数台を用いて評価環境を構築し、性能評価を行った。負荷をかけるためのソフトウェアとしては、ApacheのJmeter(version 5.3)を用いた。負荷をかける際に、メッセージを送るスレッドは100個で、10msg/sから10msg/sずつ上げながら、スループットが上限に達するまで負荷を上げた。OCBとMongoDBはDockerのコンテナを用いて構築した。登録するデータ数を変化させて、負荷をかけて性能評価をした結果を図2に示す。MongoDBに登録されているデータの数が多くなると、OCBのスループットが低減することが確認できた。この性能低下は、OCBとMongoDBがデータ連携をするときに、MongoDBにボトルネックが発生したのが原因だった。

OCBでは、クラスタ構成を採用してもスループットがスケールしない問題があると報告されており[12]、本研究で、2.2節で述べたユースケースをベースで、OCBの性能を評価した。性能評価する際に、2.2節で述べたサービスが使用するデータ量を想定し、その数のデータをMongoDBに登録しておいて評価を行った。登録するデータの数に関しては、1つのデバイスが1つのデータを送信する前提で、想定デバイス数の合計である17,000個のデータをMongoDBに登録した。評価は、単体のOCB-MongoDBの性能と、MongoDBをクラスタ構成(シャーディング)した場合の

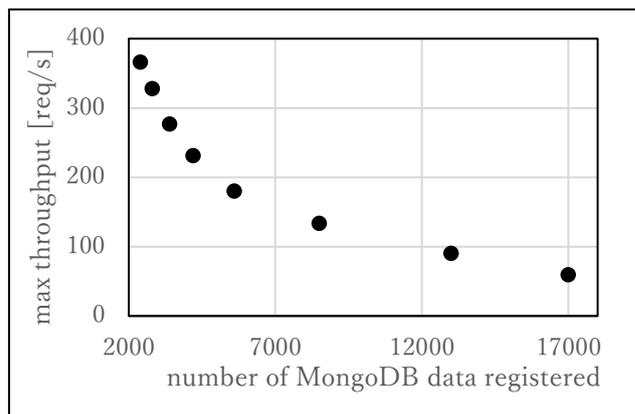


図2 MongoDBに登録されたデータ数とOCBのスループットとの関係

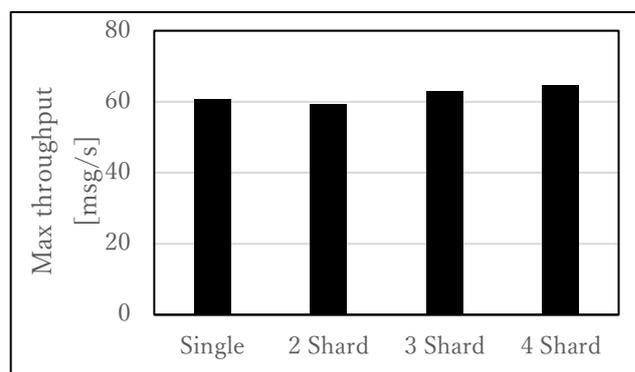


図3 MongoDBをシャーディングした場合のOCBのスループット

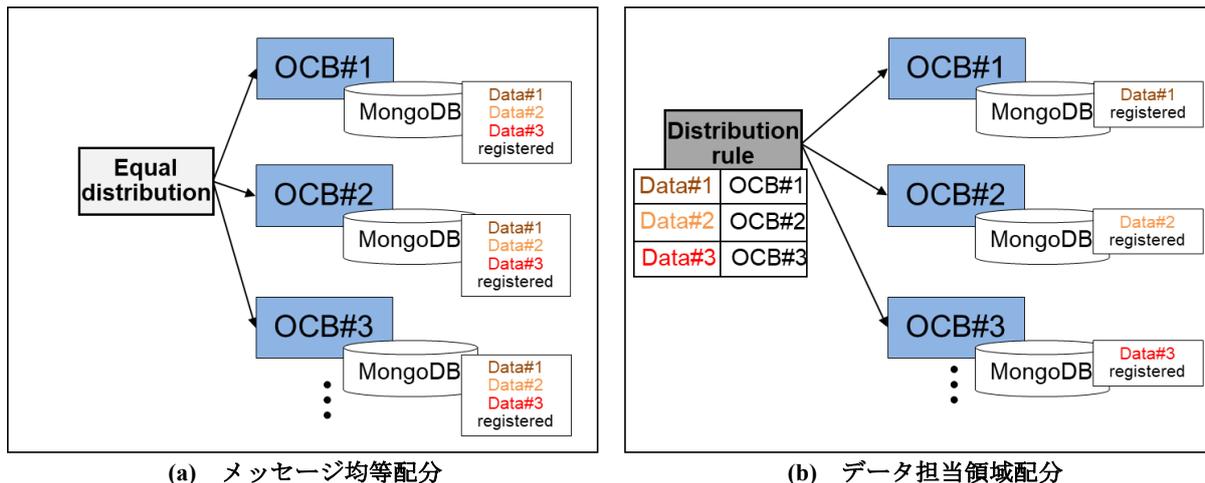
OCBの性能、二つのケースで行った。シャーディングはMongoDBのシャードを2構成から4構成まで増やして評価した。図3に単体とシャーディングの性能評価結果を示す。単体OCBのスループットは60msg/sで、シャーディングをしても、4シャード構成で64.7msg/sで、スループットが大きくスケールしなかった。シャーディングをしても、ある特定のシャードに負荷が集中し、ボトルネックが発生した原因で、スループットのスケールができなかった。この結果により、2.2節で述べていた都市OSのメッセージブローカーの要件の中で、スループットは最低限250msg/sを超えてスケールする必要があることを満たすことができなかった。

以上を踏まえて、2.3節で述べたOCBの機能を使いつつ、スループットをスケールできるシステムを実現することが課題である

3. 都市 OS のためのブローカー仮想化手法

3.1 OCBのスループットをスケールする手法

本節では、2.3節で述べたOCBの課題を解決するための手法を述べる。OCBはそのデータベースとして、MongoDBを採用しており、OCB-MongoDBは一つとして動作しているので、OCBとMongoDB間のデータやり取りを変えることは困難である。従って、データやり取りを変えない制約



(a) メッセージ均等配分

(b) データ担当領域配分

図 4 OCB 仮想化手法の二つの実現方法

の中、OCB の性能をスケールさせる方法が必要である。そこで、本研究では、独立に並べた複数の OCB を 1 つのように仮想的に扱う手法を提案する。独立に並べている複数の OCB に対してメッセージを振り分ける方法としては大きく二つが考えられる(図 4)。

一つはメッセージ均等配分で、複数の OCB に順番にまたはランダムにメッセージを振り分け、均等にメッセージが振り分けられる方法である(図 4(a))。この方法においては、メッセージがどの OCB にも転送される可能性があるため、すべての OCB は扱うデータをすべて MongoDB に登録しておく必要がある。

二つ目はデータ担当領域配分で、OCB ごとに処理するデータを予め決めて、各 OCB の MongoDB に担当のデータを登録しておいて、メッセージが来た時に適切な OCB に振り分ける方法である(図 4 (b))。振り分け先は、メッセージの中のデータ名称やデータ種類などを読み取って対応する OCB のアドレスを検索するルールを用いて決定される。二つの方法とも、OCB を増やすことでスループットをスケールできると考えられる。二つの方法の中で、どの方法が都市 OS のメッセージブローカーの要件を満たすか検討が必要である。

3.2 実現方式の比較

本節では、3.1 節で述べた方法を比較する。二つの方法が OCB のサブスクリプション・通知とデータ参照機能に対応できるかを述べて、スケーラビリティを比較する。

サブスクリプション・通知に関しては、通知でデータを取得するサービスが、すべての OCB にサブスクリプションをしておくことで、データ更新メッセージがどの OCB に転送されてもデータを受信することができる。データ参照に関しては、データ提供者がデータ更新メッセージを送信したときに、そのメッセージがどの OCB に転送されたかのアドレス情報をキャッシュなどに保存しておいて、サービスがデータ参照メッセージを送信したときに、キャッシュから参照したいデータの所在を読み取り、メッセージを

読み取ったアドレスに転送することで、サービスはデータを取得することができる。

サブスクリプション・通知に関しては、サービスが必要なデータのサブスクリプションメッセージを送信すると、メッセージの中のデータ名称・種類などを読み取って、メッセージを適切な OCB に転送し、必要なデータを担当する OCB にサブスクリプションが設定される。必要なデータの更新メッセージが送信されると、サブスクリプションが設定されている OCB に転送されるので、サービスは通知でデータを取得することができる。データ参照に関しては、データ参照メッセージを送信すると、メッセージなかのデータ名前・種類を読み取って、適切な OCB に転送されるので、サービスはデータ参照でデータを取得することができる。

上記の検討から、二つの方法とも OCB のサブスクリプション・通知とデータ参照機能に対応していることがわかる。スケーラビリティに関しては、メッセージ均等配分方法は、すべての MongoDB にすべてのデータを登録しておく必要があるため、2.4 節の図 2 で示した結果のように、MongoDB に登録されるデータの量が多くなるので、比較的に性能が低下してしまう。また、今後データが増えると、さらに性能は低下する。データ担当領域配分方法はデータの領域を分けて、担当のデータのみ MongoDB に登録すれば良いので、各 MongoDB に登録されるデータ量が少なくなるので、各 OCB が高いスループットで処理することができる。OCB を追加すると二つともスループットは向上されるが、メッセージ均等配分方法はすべてのデータが登録された OCB-MongoDB が追加されるので、データ担当領域配分方法に比べて、スケーラビリティ性が低い。

OCB 仮想化手法の比較結果、データの担当領域を分けて処理させる方が、優れたスケーラビリティを持っており、OCB の機能も活用できる。

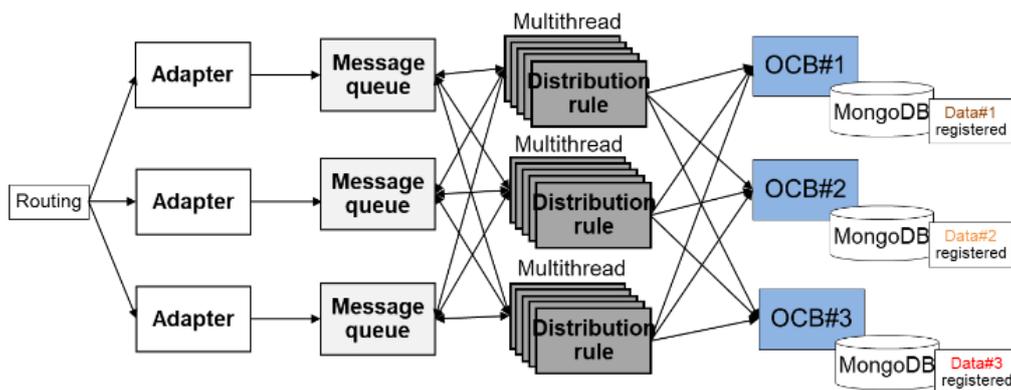


図 5 データ担当領域配分システム構成

3.3 実装方式

本節では、データ担当領域配分方法の振り分けの部分がボトルネックになることを防ぐための仕組みを述べる。提案手法では、振り分けルールの部分がシングルポイントとなり、負荷が高くなり、ボトルネックが発生する可能性がある。図 5 は図 4(b)の振り分けルール(Distribution rule)を並列に構成し、負荷を分散することでボトルネックを改組する仕組みである。振り分けルールを並列構成にしたときに、メッセージを配分する Routing はデータ提供者・利用者に公開される窓口も担う。Routing がメッセージを受信すると、メッセージ受付を担当する Adapter に送信する。Adapter は振り分けルールと繋がっているバッファとして動作するメッセージキューにメッセージを入れる。振り分けルールはメッセージキューからメッセージを取り出し処理する。このときにボトルネックが発生しないように、振り分けルールはマルチスレッド・マルチノードで構成し、複数の振り分けルールが同時に処理するようにした。マルチノードで、一つのノードに負荷が集中し負荷が高くないようにすることができ、マルチスレッドで、一つのノードに対しても処理が止まらないようにすることができる。また、メッセージキューにメッセージがたまりすぎること防ぐために、すべての振り分けルールがすべてのメッセージキューにアクセスできるようにした。

図 5 の構成にすることで、振り分けがボトルネックになり、性能が低下しないようにできる。

4. 評価

4.1 評価環境

本節では、評価環境について述べる。Figure13 に評価環境を示している。2 コア CPU, 8 GB メモリのサーバ複数台を用いて、評価環境を構築した。負荷は Jmeter を用いてかけており、メッセージを送るスレッドは 100 個で、10 msg/s から 10 msg/s ずつ上げながら、スループットが上限に達するまで負荷を上げた。メッセージの中のデータは、JSON 形式で、サイズは 1kB にした。データ数は 2.4 節と同様に 1 デバイスが 1 データを送信する前提で、17000 とし、OCB の台数を増やすと、17000 のデータを分けて登録

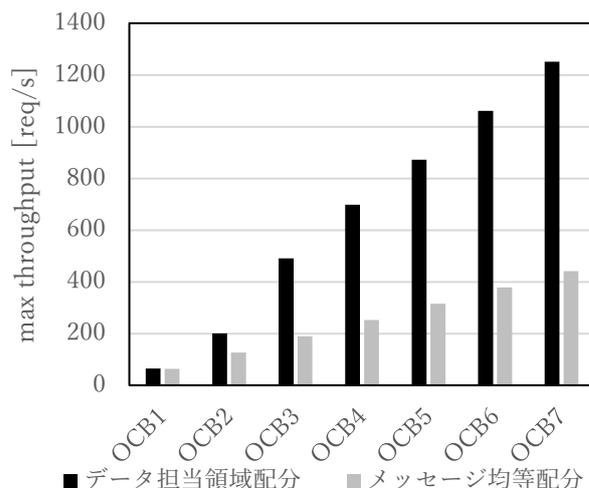


図 6 仮想化手法のスケラビリティ

して評価した。

4.2 評価結果

性能を評価した結果を図 6 に示す。図 6 の値は最大スループットで、負荷をかけたときに、ロスなくスループットが上限に達した値である。3.2 節で述べたように、データ担当領域配分方法がメッセージ均等配分方法よりスケール性があることが確認できた。OCB 台数を増やした時に、OCB1 台最大性能(60 msg/s) × 台数よりも大きくスケールすることが確認できた。これらの結果は、2.4 節で確認したように、データを分けると、MongoDB 1 台に登録されるデータ数が少なくなるので、1 台の OCB の性能がより向上したのが原因だと考えられる。OCB をデータ担当領域配分方法で、OCB3 台で、2.2 節で述べたスループットの最低要件(250 msg/s)を満たしている、OCB7 台まで増やすことで、頭打ちされずにスケールすることができた。MongoDB をシャレディングした手法と比較しても、スループットのスケール性が向上されていることがわかった。

本評価結果から、OCB 仮想化手法により、スループットをスケールできて、都市 OS のメッセージブローカーとしての要件を満たしつつ、高負荷の環境で性能低下なく、動作できることを確認した。

5. まとめ

近年, Society 5.0 の実現のために, スマートシティが注目されている. 分野横断型のスマートシティが増加している中, 分野横断的なデータ連携を容易にできるようにするためのデータ連携基盤として, 都市OS の概念が生み出された. 都市 OS の開発工数を削減するための OSS である Fiware のコアな機能であり, メッセージブローカーである OCB は性能に限界があり, 本研究で挙げたスマートシティのユースケースをもとで定めた要件を満たすことができなかった. 本研究では, Fiware のメッセージブローカー(OCB) の性能問題を明らかにし, OCB の機能を維持しながら性能向上させる課題を解決するために, メッセージブローカー機能の仮想化技術を提案した. 評価の結果, OCB の性能がスケールでき, 向上されることを確認した.

参考文献

- [1] 内閣府政策統括官, “連携によるスマートシティの将来”, 2020
- [2] 内閣府政策統括官, “スマートシティの推進に向けて”, 2020
- [3] 国土交通省都市局, “スマートシティの実現に向けて 【中間とりまとめ】”, 2018
- [4] 内閣府地方創生推進事務局, “「スーパーシティ」構想について”, 2021
- [5] 内閣府, “スマートシティリファレンスアーキテクチャ”, 2020
- [6] 日本経済団体連合会, “Society 5.0 時代の東京—デジタル革新を通じた国際競争力の強化—”, 2019
- [7] Developers Catalogue – FIWARE, <https://www.fiware.org/develop/catalogue/>
- [8] 望月康則, “世界のデータ利活用型スマートシティ開発動向”, 2018
- [9] McKinsey & Company, “スマートシティ:より快適な未来を実現するデジタルソリューション”
- [10] シスコシステムズ合同会社, “スマートシティの事例 - スマートシティがもたらす地域イノベーション”, 2016
- [11] Sanchez, Luis, et al. "Smartsantander: The meeting point between future internet research and experimentation and the smart cities." 2011 Future Network & Mobile Summit. IEEE, 2011.
- [12] Miguel Ángel Ramírez, Carlos Lucena, “Orion scalability testing report (release 5.3)”, FI-Core D.1.1.8.1, 2016.8