クラウドとIoTの統合における センサーデータ分散管理手法の提案

高木 優希[1,a)]　串田 高幸[1]

概要：
クラウドコンピューティング (以下「クラウド」と称する) と Internet of Things(以下「IoT」と称する) は異なる技術であるが, 近年, 技術の進歩とともに両者を統合する需要が高まっている. しかしながら, 両者は異なる領域であり, 別々に研究されることが多いため, 未だ標準化されたインタフェースやプラットフォームは存在していない. そこで, IoT 向けマルチクラウドソリューションを各クラウド情報と優先順位から動的に構築し, ユーザー負荷の減少及び実データの透過性向上を目的とする. 特にデータ分散管理に着目し, 実データによる検証と評価を行った. その結果, 複数クラウド使用の恩恵であるデータの処理負荷およびセキュリティ上リスクの分散が可能になった. また, この研究におけるセンサーデータおよびクラウド自動選択アルゴリズムによって, データの透過性が向上し, ユーザビリティの向上に貢献した.

1. はじめに

クラウドの考え方として, 米国国立標準技術研究所 (NIST: National Institute of Standards and Technology, 以下 「NIST」と称する) によれば, 「共用の構成可能なコンピューティングリソースの集積に, どこからでも, 簡便に, 必要に応じて, ネットワーク経由でアクセスすることを可能とするモデルであり, 最小限の利用手続きまたはサービスプロバイダとのやり取りで速やかに割り当てられ提供されるもの」である [1]. このクラウドモデルは, オンデマンド・セルフサービス, 幅広いネットワークアクセス, リソースの共用, スピーディな拡張性, サービスが計測可能であることの 5 つの基本的な特徴を持つ. また, サービスモデルとして, サービスの形で提供されるソフトウェア (Software as a Service, 以下「SaaS」と称する), サービスの形で提供されるプラットフォーム (Platform as a Service, 以下「PaaS」と称する), サービスの形で提供されるインフラストラクチャ (Infrastructure as a Service, 以下「IaaS」と称する) が存在している. さらに, 実装モデルとして, プライベートクラウド, コミュニティクラウド, パブリッククラウド, ハイブリッドクラウドが提案されている. 今日では, サービスモデルや実装モデルが多様化され, 発展を続けている. サービスモデルでは, サーバー管理をユーザーが考慮せず使用可能方向へと進んでいる. そのため, Serverless as a Service というモデルが生み出されつつある. 実装モデルでは, 単一クラウドではセキュリティ・信頼性のリスク上の観点から複数のクラウドを併用するマルチクラウドというモデルが生み出された.

日本における IoT の考え方として, 特定通信・放送開発事業実施円滑化法によれば, インターネット・オブ・シングスの実現として, 「インターネットに多様かつ多数の物が接続され, 及びそれらの物から送信され, 又はそれらの物に送信される大量の情報の円滑な流通が国民生活及び経済活動の基盤となる社会の実現をいう」と定義されている. 総務省の情報白書においても 2015 年には「自動車, 家電, ロボット, 施設などあらゆるモノがインターネットにつながり, 情報のやり取りをすることで, モノのデータ化やそれに基づく自動化等が進展し, 新たな付加価値を生み出すというもの」としている [2]. 2019 年度情報白書によれば, 自動車および輸送機器, 医療, 産業用途での IoT の高成長が見込まれる. これらはそれぞれ, コネクテッドカーの普及により, IoT 化の進展が見込まれるという点, デジタルヘルスケアの市場が拡大しているという点, スマート工場やスマートシティが拡大しているという点から予測されている.

この論文で述べるクラウドと IoT の統合の考え方は, 両者それぞれの強みを生かし, お互いの短所を補うことで, ストレージ容量や処理能力, セキュリティ上のリスクといった技術的制約を解決できるものである. IoT におけるデバ

[1] Tokyo University of Technology
Cloud and Distributed Systems Laboratory, Japan
[a)] c0117178@edu.teu.ac.jp

イスでは, ストレージや処理能力が限られているため, 全てのデータを保存することはできない. この大量のデータを保存するためには, 外部に一度保存し, そこから情報を抽出し, 演算処理をする必要がある. 一方, クラウドは, 実質的無制限のストレージと処理能力を備えた巨大なネットワークを提供している. また, 多種多様なデバイスからの動的なデータ統合を可能にする柔軟性と堅牢性も提供している. したがって, 常に単体としての機能を一切考慮せず IoT デバイスを使用することが可能になる. このような IoT とクラウドの統合は, 近年になって新しい概念として提案されるようになった. Aazam らは両者の統合およびその連係動作を Cloud of Things とした [3]. また, Botta らは統合に関する考え方を CloudIoT とした [4]. 2 章ではこのような新しい概念についての関連研究と, 一部に特化した先行研究について紹介する. 2 章によって見えてくる課題は, 3 章で詳しく記載する. 4 章では, この論文で提案するモデルについて詳しく記載する. この論文では, 外部ストレージとして各社クラウドプロバイダーの VPS 環境を利用し, データベースおよびユーザーインタフェース, 管理プログラムを構築した. 5 章では実装部分について詳しく記載する. 検証方法として, 異なるネットワーク上に設定した RaspberryPi 端末からセンサーデータを送信することを採用した. 6 章では 4 章, 5 章によって得られた結果から評価を行った. 評価方法として, データフロー把握の可否およびユーザビリティ度合いを採用した. 先行研究や既存ツールとの比較を行った. 7 章では, 3 章の課題について 6 章の評価を元に考察を行った. 最後に 8 章では結論と展望を述べた.

2. 先行研究・関連研究

この章では, この論文で述べるクラウドと IoT の統合に関して, 同様に研究を行うものについて取り上げる. まず, クラウドと IoT の統合に関する調査を行ったものについて取り上げる. Botta らはクラウドと IoT の統合について, CloudIoT という考え方とし, 統合のための課題や統合後の課題について言及した [4]. Qabil らは IoT とクラウドのアーキテクチャ及び両者を統合した概念 Cloud of things(CoT) を調査し, 問題点を言及した [5]. Biswas らはクラウドと IoT の統合における課題に対処する IoT 中心のクラウドスマートインフラストラクチャについて調査を行った [6]. Celesti らはクラウドと IoT の統合サービス IoT as a Service(IoTaaS) の進化についての考察と, 一般的な 3 層 IoT クラウドフェデレーションアーキテクチャの調査と課題について言及した [7]. Siddiqa らはビッグデータ管理に焦点を当て, ストレージ, 処理, セキュリティに重点を置き, 実現可能な管理手法を調査した [8]. 次に, 独自の開発を行ったものについて取り上げる. Persson らは記述, 接続, デプロイ, 管理の 4 つの定義に基づいた分散プラットフォーム Calvin を提案した [9]. Liu らはクラウド及び IoT データでの認証システムベースのデータ整合性検証手法に関する分析を行った [10]. Jaradat らはスマートグリッド分野におけるスマートセンサーネットワークとビッグデータの管理手法に着目し, 言及をしている [11]. Baker らは IoT におけるマイクロサービスを統合し, E2C2 と呼ばれる新しいマルチクラウド IoT サービス構成アルゴリズムを開発した [12]. Li らは IoT ソリューションプロバイダーがサービスを効率的に提供し, 継続的に拡張をするために不可欠な新しい IoT PaaS フレームワークを提案した [13]. Lea らは IoT ハブをスマートシティ PaaS として一般化するために, マルチクラウドおよびハイブリッドクラウドでの実現手法の提案を行った [14]. Sajid らはクラウドサービスを利用した IoT と物理的サイバーシステム (CPS) における重要なインフラストラクチャのセキュリティ上の課題に焦点を当て, 改善および維持するための提案を行った [15]. Jayaraman らはマルチクラウド環境の全ての IoT サービスに対してセマンティック技術を利用する階層型データ処理アーキテクチャを提案した [16]. どの論文においても, 3 章で挙げるような問題点に直面し, その一部を解決している. 本研究も同様に, 今後のクラウドと IoT の統合における問題点および課題の解決に貢献することを期待する.

3. 課題

この章では, 2 章で挙げた先行研究・関連研究および調査にしたがって, この論文で取り上げる現状の課題を 4 つ挙げる.

3.1 ストレージ

多くのセンサーデータを保存するためには, ストレージ容量も大きくなければならない. IoT デバイスにはユーザーがストレージを追加しない限りはストレージ容量の増加はできない. 一方, クラウドではこの考えは必要ではない. クラウドサービスプロバイダーがストレージを管理するため, ユーザーの観点からすれば, ストレージ容量は実質無制限である. Amazon Web Service(以下「AWS」と称する) で導入されている EBS のように, ブロック単位でストレージを管理する, あるいは自動的にスケーリングする仕組みができれば良い [4][6].

3.2 リソースの割当

IoT デバイスから送信されたデータがクラウド上のリソースを要求する場合, リソースの割り当ては課題になる [3]. どの IoT がどのリソースをどれだけ必要とするかを決定するのは非常に難しいからである. センサーとセンサーの使用目的, データ生成のタイプ, 量, 頻度に応じて, リソースの割り当てを自動的に決定する必要がある. 現在, これらは手動で行われていることが多く, ユーザーへの負

荷が増加する. 単一のクラウドサービスあるいは, 複数のクラウドで構成されるマルチクラウドの中で, 自動的にデータを管理するシステムが必要とされるのである.

3.3 スケーラビリティと柔軟性

IoT が利用される場合の中には, リアルタイムイベントが発生するものがある. IoT デバイスによって収集されたデータとそれに沿った処理を適切なサービスやアプリケーションに反映させる必要がある. リアルタイムイベントでは, ユーザーがリクエストを行ってから, 処理が行われ, 反映がされるまでは速度が求められる. つまり, 効率的なアルゴリズムやメカニズムが必要となる [4]. また, クラウドを使用する以上, 柔軟なサブスクリプションと設定管理を提供する仕組みが必要である. 支払う金額によって信頼性が異なるため, 通常はスケーラビリティを保証することが難しくなる.

3.4 信頼性

現在, サーバーレス化が進んでいる. データの信頼性はクラウドサービスプロバイダーに依存するケースが多い. ユーザーがデータを管理するのではなく, クラウドサービスプロバイダーに一任するのである. 従って, オンプレミスやハイブリッドクラウドの場合と比較し, データ保証や透過性といった観点から信頼性が低下する [6]. また, 実データをクラウドに集約するほど, 透過性がなくなり, 管理における信頼性が低下する.

4. 提案モデル

この章では, この論文で提案するモデルおよび構成について述べる. まず, 全体的な構成としては, センサーデバイスから得た情報を Web サーバへ送信することで最適なクラウドデータベースへ格納する. センサーデバイスの数や格納先であるクラウドの種類は実質無制限である. センサーデータが Web サーバに送信される際には優劣を付けることで, クラウドの選択を容易に実現している. 中間層となっている Web サーバがセンサーや各クラウドの設定を管理するため, IoT およびクラウドを操作する負担が大いに減少されるのである. 次に, クラウドの選択手法として, 各クラウドから一定期間ごとにマシン情報を取得し, その情報を元に優先順位付けを行う. 詳しい手順は 5 章で示す. この優先順位の高いクラウドから順にセンサーデバイスを割り当てる. このようにすることで, 単一のクラウドに偏ることなくセンサー情報に従って分散管理が可能となる. また, どのセンサーデータがどのクラウドに格納されているかが明確に把握できるというメリットがある. その次に, データの仕訳モデルとして, 2 つ提案を行う. 第一に, ラベリングを行う手法について提案を行う. ラベリングモデルでは, センサーデバイス上と Web サーバー上にラベリングの設定ファイルを設置する. 一定期間ごとに同期を行うことで, 設定を統一化させる. これによって, ラベルの付け方に関して IoT デバイス側と Web サーバ側で整合性を保つことが実現可能である. 第二に, 機械学習を用いた手法について提案を行う. k-近傍法によるデータ分類を行う. ラベリング手法では, 必ず期待通りの分類が可能な反面, ユーザー入力を必要としている. このユーザー入力を無しで行うことを可能とするのが, 機械学習を用いた手法である. しかしながら, 分類の精度はデータ量に依存するため, 一定期間ごとに学習を繰り返す必要がある. それぞれの特徴を表 1 に示す. 表 1 では, ラベリングモデルと機械学習モデルそれぞれの機能について比較を行った. 両者ともデータをユーザーが手動で分類する必要はない. ラベリングモデルでは, ラベルに従って分類されるため, 必ず期待通りとなる. しかし, IoT デバイス側でデータに対してラベル付けを行うため, 初期設定としてユーザーに入力処理を求める必要がある. さらに, Web サーバと IoT デバイス間でラベルデータを統一化する必要があるため, 同期が必要である. 機械学習モデルでは, 学習が必要となるため, 事前データが必要である. しかし, ユーザーが入力を行う必要や同期を行う必要はない.

機能	ラベリングモデル	機械学習モデル
自動分類	○	○
分類精度	○	△
事前データ不要	○	×
入力不要	×	○
同期不要	×	○

表 1 データ仕訳提案モデルの比較

5. 実装

5.1 実験環境

本研究で使用した IoT Device, Web Server, Cloud についてそれぞれ記す. この研究では, 各種センサーデータで検証実験を行うため, 単一のコンピュータで複数のセンサーを付けることが可能な Raspberry Pi を IoT デバイスとする. センサーには温度, 湿度, 気圧, 高度といった一般的なセンサー情報を取得できるものを採用した. OS では, パッケージ管理のし易さを重視し, Linux の Debian 系ディストリビューションである Ubuntu や Raspbian を採用した.

IoT Device

Raspberry Pi 3 model B+と Raspberry Pi Zero を使用した. OS は Raspbian10.2 を採用, センサーには ADT7410, DHT11, BMP085 を使用した. また, センサーから値を取得及びデータ送信に使用するプログラムは Python3.7.3 を使用している.

Web Server

Google Cloud Platform(以下「GCP」と称する) の Computer Engine を利用し, VM インスタンスを作成した. OS は Ubuntu 18.04 LTS を採用し, データを送受信するプログラムは PHP7.2.24 を使用している.

Cloud(External Storage)

クラウドサービスとして主流とされる GCP, AWS, Microsoft Azure(以下「Azure」と称するを使用した. 実験環境では, データベースは全て MySQL を使用した. GCP は Web Server 同様に VM インスタンスを作成し, OS は Ubuntu 18.04 LTS を採用した. AWS では 2 パターンを作成した. 第一に, Amazon EC2 において GCP と同様に VM インスタンスを作成し, OS は Amazon Linux 2 を採用した. 第二に, Amazon RDS で MySQL データベースを作成した. Azure も同様に 2 パターンを作成した. 第一に, Virtual Machine から OS が Ubuntu 16.04 LTS の VM インスタンスを作成した. 第二に, Azure Database for MySQL を利用し, データベースを作成した.

5.2 IoT デバイス側の処理

IoT デバイス側では, ユーザーが IoT デバイスに各種センサーを取り付けた後, センサーの種類が何かをユーザーが入力を行う. センサーの種類として入力できるものは, センサーデータとして一般的である温度, 湿度, 気圧, 高度, その他の 5 種類とした. 入力は対話方式であり, その情報は別ファイルに保存された後, Web サーバへ送信され, Web サーバ側のデータベースに保存される. 入力は具体的なフローを図 1 に示す. ユーザーからの入力情報を元に, ルールに従いそれぞれの情報に優先順位を付与する. ルールについての詳しい情報を以下に示す.

ルール 1

Raspberry Pi に接続されているセンサーの数が多いほど, そのセンサーの種類 (温度, 湿度, 気圧, 高度, その他) の優先度が高くなる

ルール 2 : Priority of Sensor data

1. Temperature
2. Humidity
3. Altitude
4. Barometric pressure
5. Others

ルール 3 : Priority for Storing (DB)

1. Available Storage
2. Storage Use %
3. Memory
4. CPU cores

IoT デバイス側ではルール 1・2 が適用される. ルール 2 では, クラウドの選択を容易にするため事前に優先度を付与している. この研究では, 使用した 3 種類のセンサーから取得が可能なセンサー情報の多い順としている. この優先度は IoT デバイス側の設定ファイルおよび Web サーバ上のデータベースによって管理されている. IoT デバイス側でセンサーの種類についての対話式の入力 (初期設定) を行う際に, 情報を変更することで, センサーの種類は変更可能である. そのため, 柔軟な運用が可能である. Web サーバに送信された優先度とセンサーの対応付けの情報は, クラウドの選択に利用される. つまり, 優先度の高いセンサーデータから順に最適なクラウドが選択されていくということである. ルール 3 については, クラウド側の処理で詳しい説明を行う.

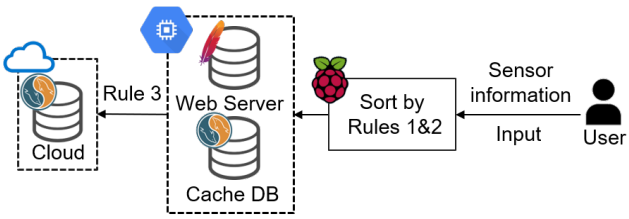

図 1 IoT デバイスからクラウドまでのデータフロー

5.3 クラウド側の処理

図 1 のように, IoT デバイスから送信されたセンサー情報は, キャッシュ用のデータベースに保存される. また, 各外部ストレージから一定期間ごとにマシン情報を送信させ, キャッシュ用のデータベースに保存する. これらの情報を元に, ルール 3 に基づいてどのクラウドが最適かを判断する. ルール 3 は, データ保存に関する情報ほど優先度を高く設定している. センサーデータは蓄積され, 膨大なデータとなることが予想されるため, ストレージに関わるものほど優先度が高い. つまり, マシン情報の最終更新時の状態で最もストレージ容量に十分な空きがあるものが選択される. この時, 各センサー情報とクラウドの対応表がデータベース内で作成され, 以降のセンサーデータ受信時に使用される. 対応表が作成され, クラウドが割り当てられた後, IoT デバイス側からデータを受信するようになる. この時, 現在時刻から前後 1 時間の範囲で 5 日間分のデータを取得し, 最大値と最小値を閾値とする. 日本には四季があり, 1 年を通して気温や湿度, 気圧は変動がしやすい. さらに地球温暖化の影響もあり, 日ごとの値の変動が大きい. その変化に対応するため, 短い間隔で更新を行うと同時に, 精度を向上させるため, 10 個以上のデータが必要であった. 加えて, 数回のうち異常値が発生した場合を考慮し, 5 日間分の毎時間ごとのデータとした. 正常値であれば保存用のテーブルに格納される. 異常値であれば, 例外用のテーブルに格納され, 監視サービス Mackerel を通して Slack に通

知が送信される．その後，2 種類の手法によってデータが分類され，各データに合った外部ストレージ (VPS あるいは SQL) へ送信される．

5.4 ラベリングモデル

ラベリングモデルでは事前にユーザーが IoT デバイス側でプログラムを実行し，ラベルの設定を行う．ここでのラベルとは，ルール 2 のように，センサーの種類に合わせて割り振る番号のことである．また，設定とはそれぞれのセンサーごとにラベルを割り当てることである．具体的には，ユーザーがラベルの設定ファイルに任意の値を書きこむということである．その後，センサーデータ取得のプログラムを実行することで，センサーデータとラベルを一緒にして送信する．具体的な構成を，図 2 に示す．整合性を維持するために，IoT デバイス側では cron を実行し，データを受信するサーバ側およびデータを送信する IoT デバイス側の両者でラベルデータを 1 日に 1 回同期する．

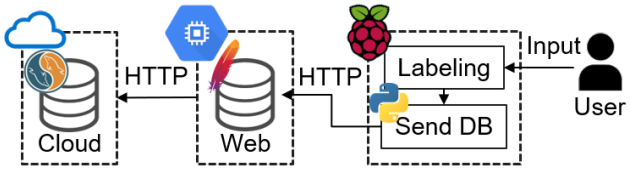

図 2　ラベリングモデル

5.5 k-近傍法による分類モデル

具体的な構成を図 3 に示す．k-近傍法による分類モデルでは，ラベリングモデルのようにラベルと一緒に事前に学習を行う．教師あり学習により，新しいデータが入力された際にどのラベルに近いかを予測させるためである．データは特徴量化され，パラメータとして与えられる k 個の最近傍が標本として選択される．この k 個の標本のラベルの多数決が行われ，結果が出力される．具体例を図 4 に示す．

図 3　k-近傍法による分類モデル

これを使用し，送信されたセンサーリソースの分類を行う．この研究ではプログラミング言語を Python，ライブラリを scikit-learn で k-近傍法モデルを構築した．標本数 k にあたるパラメータは n_neighbors であり，この研究では 3 としている．これは，センサーの種類ごとに値の差が大きく，かつ分散しにくいため，標本数を小さく設定することで，間違った判定をしないようにするためである．ラベリングモデルと同様に，分類が終了したセンサーデータは適切なクラウドへ送信される．

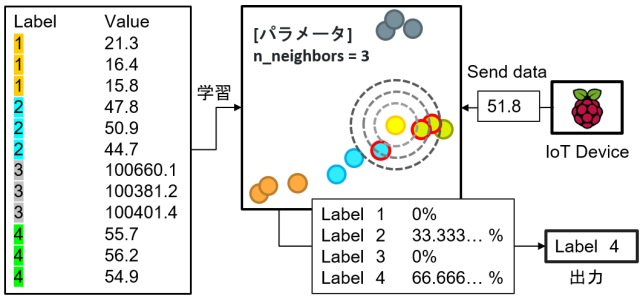

図 4　k-近傍法の例

6. 評価

実験の検証および評価として，既存のプラットフォームやアプリケーションを使用し，提案モデルと同様のデータ送信およびデータ管理を行った．その際のユーザービリティについて，手順の数や学習にかかった時間から，導入や変更のしやすさについて評価を行った．提案モデルでは，ラベリングモデルのような，事前にデータセットを用意する場合は，IoT デバイスを利用するユーザーに対して多少の手間をかけることになるが，必ず正確な仕訳が可能である．k-近傍法による分類では，ユーザーはデータを Web インタフェースに対して送信するだけで良い．しかし，分類は学習量に依存するため，必ず正しい仕訳が成されるわけではない．この研究での実験では，実データを 1 時間毎にとっているためデータ数が不足している状況ではあったが，各センサーデータの値が分散していることで失敗はなかった．実際のビッグデータとして扱う場合，精度が向上すると予想される．この研究で作成したモデルと GCP, AWS, Azure のようなクラウドプロバイダーが用意している既存の IoT プラットフォームとの比較を表 2 に示す．

ユーザビリティ	提案モデル	既存モデル
自動分類	○	×
マルチクラウド	○	△
学習コストの低さ	×	○
透過性	○	△
柔軟性	△	○
拡張性	○	○
GUI	△	○

表 2　データ仕訳提案モデルの比較

この研究では，自動分類を可能とした点については最も既存モデルに勝っているといえる．さらに，データの格納場所をユーザーが任意で決定し，把握できるという透過性の点についても勝っているといえる．既存モデルでは，データの流れには着目せず，データそのものに対して監視を行

うからである．また, IoT 向けプラットフォームとして, マルチクラウドに対応させている点についても既存モデルより勝っている．これは, クラウドを複数構成する際に, 提案モデルのほうが遥かに自由度が高いからである．しかし, 自由度が高いといってもすぐに変更が反映されない点やクラウドの数が増加する際は設定ファイルを書き換えなければならない点といった柔軟性, 学習コストの高さに課題が残っているといえる．また, クラウドプロバイダーの GUI は機能が豊富かつ綺麗である．これらの点が課題として挙げられるが, 3 章で述べた課題に関しては改善されているといえる．次の 7 章で詳しく説明を行う．

7. 考察

この章では, 3 章で述べた問題点に対し, 5 章で述べた結果と照らし合わせ, 考察を述べる．この研究の実験により, 3 章の課題が改善された．

7.1 ストレージ

マルチクラウド環境を構築することで, 実質的無制限のストレージ容量が成立した．ユーザーが各クラウドのストレージ容量を増設あるいは, マルチクラウド構成におけるクラウド数の増加を行うことで, ストレージ容量がなくなることがはない．また, IoT デバイスをクラウドに接続することで, ユーザーは IoT デバイス上のストレージ容量を考慮する必要がなくなる．この研究においても, データの格納先はクラウドストレージであり, ローカル環境に格納先を造る必要がないのである．また, IoT はクラウドとの統合により, ストレージ容量だけでなく, 演算処理をはじめとする IoT デバイス単体ではボトルネックとなっていた問題が解消される．IoT が進化を続けた先にはクラウドとの統合があるといえる．

7.2 リソースの割当

この研究ではクラウドの選択がリソースの割り当てに直結している．そのため, 柔軟性の高い自動的なリソースの割り当てが実現できた．これは, マルチクラウドだけでなく, ハイブリッドクラウドにおいても実現が容易である．この研究では大まかなストレージに対するリソースの割り当てを行ったが, 単一のリソースをさらにパーティションで区画し, 最適化するアプローチと組み合わせることで, より効果を発揮する．大まかな部分から詳細までユーザーに選択しを与えることでユーザビリティの向上につながるといえる．

7.3 スケーラビリティと柔軟性

マルチクラウド環境において, 単一クラウドより柔軟なスケーラビリティが得られた．さらに, 各クラウド単体においてもスケーラビリティと柔軟性は十分であり, さらにこれらを複数管理し, 制御することでユーザーの期待値以上の処理が実現可能となる．例えば, オートスケーリングやこの研究で行ったような自動的なクラウド選択, あるいはクラウドの自動切換えである．このような高いスケーラビリティと柔軟性は, ローカル環境がボトルネックとなるオンプレミスやハイブリッドクラウドでは実現できないものである．今後もマルチクラウドはアプリケーションやプラットフォーム開発における進化に貢献するといえる．

7.4 信頼性

この研究の様に, 大手クラウドプロバイダーのサービスを複数利用することで, サービス品質が相対的に保証され, データ管理における信頼性が向上した．また, 提案したデータの分散管理手法での透過性の向上により, データの整合性の観点からも信頼性が向上した．さらに, 提案したモデルとマルチクラウドにおけるバックアップアプローチを組み合わせることで, サービス品質およびデータの保守性が向上できるといえる．

8. おわりに

最後に, これまでの章をまとめ, 結論を述べる．この研究での環境下では, 3 章で述べた課題は解消された．IoT デバイスとクラウドの間に中間層として Web インタフェースを構築することで, リソースの最適化や割り当てに関する問題や柔軟性, 拡張性, 信頼性といった点が解消されることがわかる．しかしながら, 中間層を設けることによって, ネットワーク遅延やオーバヘッドといったクラウドサービスプロバイダーに依存する問題点が解消できないという点がある．IoT とクラウドそれぞれの技術のみで解決するのではなく, 両者を共に考慮した際に起こりうる問題に取り組むべきである． この論文の試みにより, クラウドと IoT の統合におけるストレージに関わる課題が解決された．複数のストレージを管理し, 最適化することで従来のオンプレミス環境での運用より遥かにデメリットの少ない運用が可能になる．この研究で使用したユーザーインタフェースのグラフィカル面での向上や管理の単純化を向上させることで, さらにユーザビリティを改善させることができると考えられる．透過性については, クラウドを使用する以上, データの監視やデータフローの可視化といった別の手法を使用する以外に向上させることはできない．そのため, 透過性の向上には新規プラットフォームの開発が必要となる．各技術の学習コストの低下あるいは考慮せず使用可能なプラットフォームを完成させることを最終目標として改善を行う．

参考文献

[1] Mell, P. and Grance, T.: The NIST Definition of Cloud Computing, Technical report, National Institute of Stan-

dards and Technology (2011).
[2] Anonymous: WHITE PAPER, Ministry of Internal Affairs and Communications, Japan (2015).
[3] Aazam, M., Khan, I., Alsaffar, A. A. and nam Huh, E.: Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved, *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences and Technology (IBCAST) Islamabad, Pakistan, 14th - 18th January, 2014* (2014).
[4] Botta, A., Donato, W., Persico, V. and Pescapé, A.: Integration of Cloud computing and Internet of Things: A survey, *Future Generation Computer Systems*, Vol. 56, pp. 684–700 (2016).
[5] Qabil, S., Waheed, U., Awan, S. M., Mansoor, Y. and Khan, M. A.: A Survey on Emerging Integration of Cloud Computing and Internet of Things, *2019 International Conference on Information Science and Communication Technology (ICISCT)* (2019).
[6] Biswas, A. R. and Giaffreda, R.: IoT and cloud convergence: Opportunities and challenges, *2014 IEEE World Forum on Internet of Things (WF-IoT)* (2014).
[7] Celesti, A., Fazio, M., Giacobbe, M., Puliafito, A. and Villari, M.: Characterizing Cloud Federation in IoT, *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)* (2016).
[8] Siddiqa, A., Hashem, I. A. T., Yaqoob, I., Marjani, M., Shamshirband, S., Gani, A. and Nasaruddin, F.: A survey of big data management: Taxonomy and state-of-the-art, *Journal of Network and Computer Applications*, Vol. 71, pp. 151–166 (2016).
[9] Persson, P. and Angelsmark, O.: Calvin – Merging Cloud and IoT, *Procedia Computer Science*, Vol. 52, pp. 210–217 (2015).
[10] Liu, C., Yang, C., Zhang, X. and Chen, J.: External integrity verification for outsourced big data in cloud and IoT: A big picture, *Future Generation Computer Systems*, Vol. 49, pp. 58–67 (2016).
[11] Jaradat, M., Jarrah, M., Bousselham, A. and Ayyoub, M. A.: The Internet of Energy: Smart Sensor Networks and Big Data Management for Smart Grid, *Procedia Computer Science*, Vol. 56, pp. 592–597 (2015).
[12] Baker, T., Asim, M., Tawfik, H., Aldawsari, B. and Buyya, R.: An energy-aware service composition algorithm for multiple cloud-based IoT applications, *Journal of Network and Computer Applications*, Vol. 89, pp. 96–108 (2017).
[13] Li, F., Voegler, M., Claessens, M. and Dustdar, S.: Efficient and Scalable IoT Service Delivery on Cloud, *2013 IEEE Sixth International Conference on Cloud Computing* (2013).
[14] Lea, R. and Blackstock, M.: City Hub: A Cloud-Based IoT Platform for Smart Cities, *2014 IEEE 6th International Conference on Cloud Computing Technology and Science* (2014).
[15] Sajid, A., Abbas, H. and Saleem, K.: Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges, *IEEE Access*, Vol. 4 (2016).
[16] Jayaraman, P. P., Perera, C., Georgakopoulos, D., Dustdar, S., Thakker, D. and Ranjan, R.: Analytics - as - a - service in a multi - cloud environment through semantically - enabled hierarchical data processing, *Software : Practice and Experience*, Vol. 47, No. 8 (2016).