3J-01

ゲートウェイクラウドを用いたサービス指向 IoT アーキテクチャの提案

濱野　真伍†　　久間　一輝†　　青山　幹雄†
南山大学　情報理工学部　ソフトウェア工学科†

1. 研究背景と課題

IoT の進展によってインターネットに接続されるデバイスの数が飛躍的に増大するため，スケーラビリティが高いアーキテクチャが必要となる．本稿ではサービス指向に基づく，スケールアウト可能な IoT アーキテクチャを提案する．

2. 関連研究

2.1. oneM2M の機能アーキテクチャ

oneM2M が提案する IoT の機能アーキテクチャは，インフラサーバ，ミドルノード，デバイスノードの 3 層からなる[3, 4]．しかし，スケールアウトについては明確はない．

2.2. エッジ/フォグコンピューティング

クラウドとデバイスの間にゲートウェイなどの処理を行うサーバを配置するアーキテクチャである[1]．

3. アプローチ

本稿では，oneM2M の機能アーキテクチャの構成に基づきインフラサーバとミドルノードの間にゲートウェイクラウドを設置することによりスケールアウト可能な IoT アーキテクチャを提案する．

4. 提案アーキテクチャ

4.1. ゲートウェイクラウドによるスケールアウトの実現

ゲートウェイクラウドではデバイスから送信された計測データを Publish/Subscribe ブローカによりインフラサーバに配信する．これにより，デバイスとインフラサーバは互いの状態に関わらず，データを送受信できる．また，接続されるデバイスの数が増加した場合にゲートウェイクラウドでスケールアウトすることで対応する．提案アーキテクチャを，論理アーキテクチャ，物理アーキテクチャ，配置アーキテクチャで定義する．

4.2. 論理アーキテクチャ

論理アーキテクチャを図 1 に示す．複数のデータ計測機能で計測されるデータをデータ送信機能が送信し，複数のデータ受信機能のサブスクリプションに従って計測データが受信される．すべてのデータを受信するデータ受信機能，データ登録機能によって，計測データをデータベースに登録することでデバイスに直接アクセスせずに計測データを収集できる．さらに，送信先分配機能によってデータの送信先を分配しスケールアウトを可能にする．

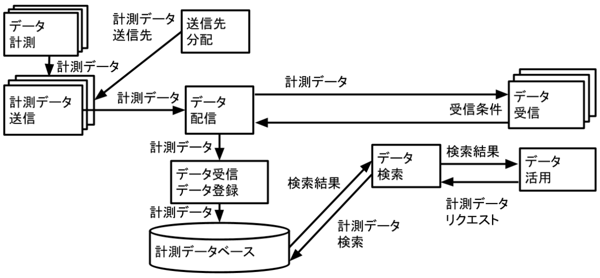

図 1 論理アーキテクチャ

4.3. 物理アーキテクチャ

物理アーキテクチャを図 2 に示す．本アーキテクチャは膨大な数のデバイスに対応するために，ゲートウェイクラウドをスケールアウトする構成を提案する．1 つのゲートウェイクラウドに接続されるデバイス数はそのクラウドの処理能力によって決定する．

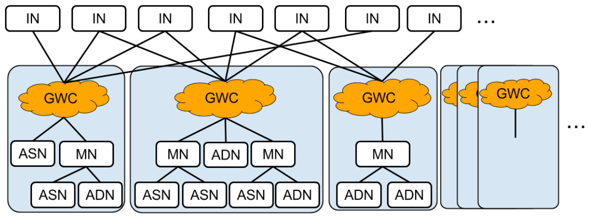

図 2 物理アーキテクチャ

4.4. 配置アーキテクチャ

配置アーキテクチャを図 3 に示す．

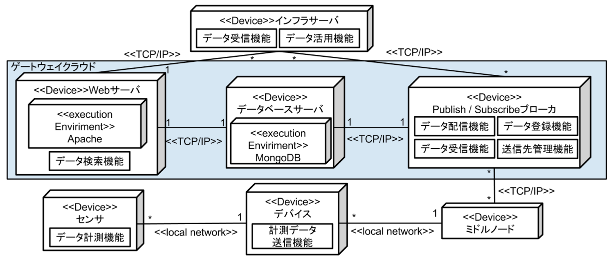

図 3 配置アーキテクチャ

ゲートウェイクラウドノードに Web サーバ，データベース，Publish/Subscribe ブローカを配置する．また，Publish/Subscribe アーキテクチャに基づき，本アーキテクチャでは，デバイス，ミドルノードを Publisher として扱い，インフラサーバを Subscriber とする．ゲートウェイクラウドに Publish/Subscribe ブローカを配置することにより，インフラサーバとデバイスノードの間を非

A Service-Oriented IoT Architecture with Gateway-Cloud
†Shingo Hamano, Kazuki Kuma, Mikio Aoyama, Department of Software Engineering, Faculty of Information Sceince and Engineering, Nanzan University

同期かつ独立となるように仲介する．そのため，デバイスに対してインフラサーバから直接通信が行われることはない．

5. プロトタイプの実装

(1) プロトタイプにおけるゲートウェイクラウドとデバイスの実装環境を表 1，表 2 に示す．

表 1 ゲートウェイクラウドの実装環境

コンポーネント	仕様
OS	Ubuntu 14.04 LTS
CPU	Intel® Core™2 Duo CPU E7300 @ 2.66Ghz×2
メモリ	2GB
Publish/Subscribe サーバ	Mosquitto 1.4.3[2]
Web サーバ	Apache 2.4.7
データベース	MongoDB 3.0.7

表 2 デバイスの実装環境

コンポーネント	仕様
マシン	Raspberry Pi B+
温度センサ	MCP9700 -E/TO
コンバータ	MCP3008-8Channel 10Bit

(2) プロトタイプの構成を図 4 に示す．

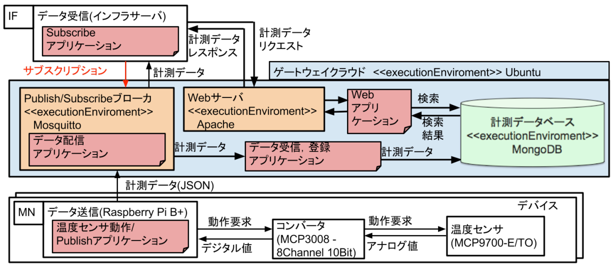

図 4 プロトタイプの構成

6. プロトタイプの実行

プロトタイプの実行シナリオは Raspberry Pi に接続した温度センサの計測データをゲートウェイクラウドに送信し，計測データベースへの登録とインフラサーバでの受信を確認する．Raspberry Pi からの送信周期は処理能力の限界値である 6 ミリ秒ごととする．また，インフラサーバから REST を用いて，ゲートウェイクラウドのデータベースから任意のデータを受信可能であることを確認する．Raspberry Pi の計測データを図 5 に示し，インフラサーバの受信結果を図 6 に示す．

```
2016-01-06 13:59:03.041
temp :    23.55
2016-01-06 13:59:03.047
temp :    23.87
2016-01-06 13:59:03.053
temp :    25.81
2016-01-06 13:59:03.058
temp :    23.87
```

図 5 Raspberry Pi における計測データ

```
tempture/room1 {"date": "2016-01-06 13:59:03.041", "D_id": "D01", "id": "D01_121", "temp"
: 23.55}
tempture/room2 {"date": "2016-01-06 13:59:03.044", "D_id": "D02", "id": "D02_19", "temp":
 30.0}
tempture/room1 {"date": "2016-01-06 13:59:03.047", "D_id": "D01", "id": "D01_122", "temp"
: 23.87}
tempture/room2 {"date": "2016-01-06 13:59:03.052", "D_id": "D02", "id": "D02_20", "temp":
 29.68}
```

図 6 インフラサーバにおけるデータ受信状況

図 5 と図 6 のデータが一致していることからデバイスで生成されたデータがインフラサーバで受信できていることが確認できる．また，同じタイミングで生成したデータを保存したデータベースを図 7 に示す．

```
> use admin
switched to db admin
> db.temp_test.find()
       中略
{ "_id" : ObjectId("568c9f59b1e6d6c036b2fc02"), "temp_id" : "D01_121", "topic" : "temptur
e/room1", "date" : "2016-01-06 13:59:03.041", "tempture" : 23.55, "Device_id" : "D01" }
{ "_id" : ObjectId("568c9f59b1e6d6c036b2fc03"), "temp_id" : "D02_19", "topic" : "tempture
/room2", "date" : "2016-01-06 13:59:03.044", "tempture" : 30, "Device_id" : "D02" }
{ "_id" : ObjectId("568c9f59b1e6d6c036b2fc04"), "temp_id" : "D01_122", "topic" : "temptur
e/room1", "date" : "2016-01-06 13:59:03.047", "tempture" : 23.87, "Device_id" : "D01" }
{ "_id" : ObjectId("568c9f59b1e6d6c036b2fc05"), "temp_id" : "D02_20", "topic" : "tempture
```

図 7 MongoDB に保存されたデータ

図 6 に示すように配信と同時にデータベースへの保存も完了している．次に，REST を用いた要求/応答型のデータ収集例を図 8 に示す．GET メソッドを用いることで，計測データベースに保存してあるデータから，必要なデータをリソースとして取得できる．

```
10.48.134.218/data_search.php?if=4&kind=tempture&max=22.0&min=20.0

temptureが20.0以上22.0以下のものを表示します

array(6) {
 ["_id"]=>
 object(MongoId)#7 (1) {
  ["$id"]=>
  string(24) "568c9f58b1e6d6c036b2fb98"
 }
 ["temp_id"]=>
 string(6) "D01_34"
 ["topic"]=>
 string(14) "tempture/room1"
 ["date"]=>
 string(23) "2016-01-06 13:59:02.546"
 ["tempture"]=>
 float(20.65)
 ["Device_id"]=>
 string(3) "D01"
}
```

図 8 REST を用いた計測データ受信例

7. 評価

(1) Publish/Subscribe ブローカによるスケーラビリティの評価

Publish/Subscribe ブローカを用いることでデバイスからのイベントを待ち合わせなく，発生順にインフラサーバで受信可能になった．

(2) ゲートウェイクラウドの計測データ収集の評価

Publish/Subscribe ブローカと 要求/応答型の REST のサーバをゲートウェイクラウドで実装することで，インフラサーバはデバイスデータのリアルタイムな収集と，データベース検索によるタイミングによらない収集が可能になる．その結果，計測データの収集の自由度が高まり，インフラサーバの計測データ収集の完全性を保証する．

8. まとめ

膨大な数のデバイスに対応し，ゲートウェイクラウドと Publish/Subscribe ブローカによるスケーラブルな IoT アーキテクチャを提案した．プロトタイプにより，アーキテクチャの有用性を示した．

参考文献

[1] F. Bonomi, et al., Fog Computing and Its Role in the Internet of Things, Proc. of MCC '12. pp. 13-16.
[2] Mosquitto, http://mosquitto.org/.
[3] oneM2M, Functional Architecture, Jan. 2015.
[4] 富田二三彦 他(編), M2M/IoT 教科書, インプレス, 2015.