IoT for Aquarium Project

High level design

Empowering Creativity, Delivering Excellence

Vietnam DFT Technology Joint Stock Company

Office: Kim Anh Building, Lane 78 Duy Tan, Cau Giay, Hanoi, Vietnam

HISTORY

Version No.	Issue Date	Status	Reason for Change
1.0	29/03/2023		ドキュメントのフレームワーク
2.0	30/03/2023		アーキテクチャの補足
3.0	31/03/2023		内容の完成

REVIEW

Reviewer's Details	Version No.	Date
Nguyễn Tuấn Anh (CTO)	1.0	30/03
Nguyễn Tuấn Anh (CTO)	2.0	31/03
Nguyễn Tuấn Anh (CTO)	3.0	01/04

目次

図の目次

1 一般的な紹介

農業は長い間、農家の経験に大きく依存しており、より効率的な栽培・養殖方法を見つけることが大きな課題となっています。第4次産業革命の時代において、IoTは最適かつ不可欠な技術であり、新しい技術を生産および栽培活動に適用することがトレンドとなっています。IoTは、農業を定性的な生産分野から、収集、統合、および統計分析に基づいた正確な生産分野へと変えることができます。

Figure 1: IoTダッシュボード

2 技術ソリューションの提案

2.1 ソリューションの概要

システムの論理アーキテクチャ

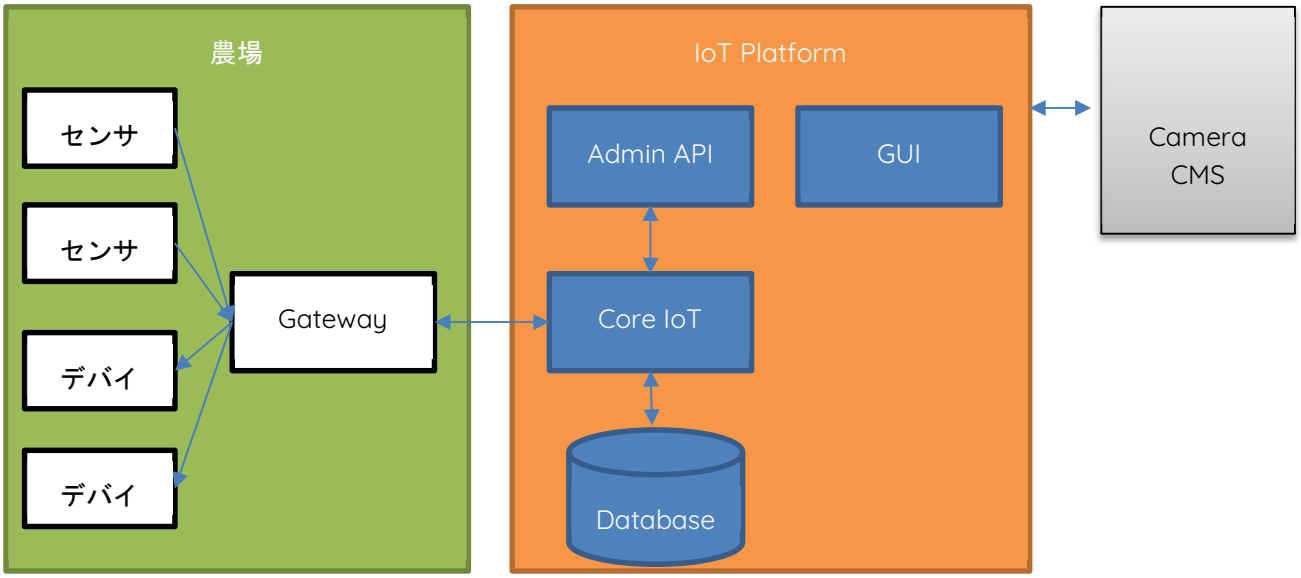

Figure 2: 論理アーキテクチャ

物理接続アーキテクチャ:

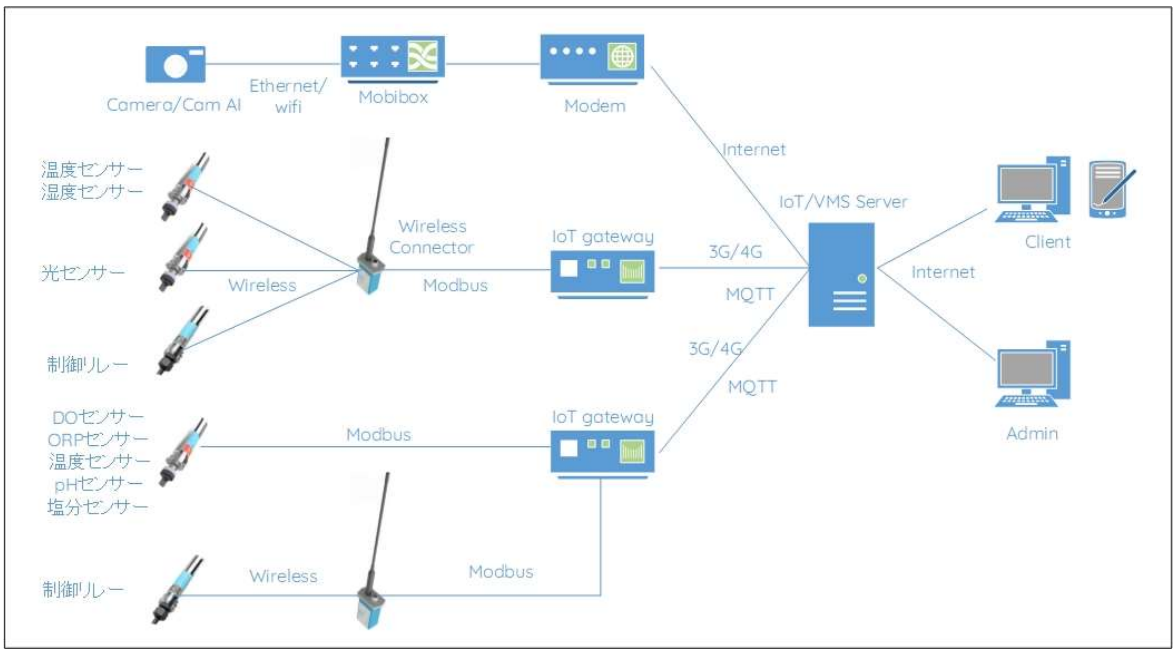

Figure 3: 展開モデル

ソフトウェア管理・監視のためのIoTプラットフォームは、以下のコンポーネントで構成されています。すべてのコンポーネントはオープンスタンダードまたはオープンソースプラットフォームであり、ライセンス費用は不要です。

- IoTプラットフォーム（Thingsboardオープンソースプラットフォームを使用）：デバイス管理、デバイス接続、センサーデータ、デバイス設定、警告、デバイスステータスなどのAPI（アプリケーションプログラミングインターフェイス）を提供します。
- Admin API：サードパーティのアプリケーションが使用するAPIを提供します。これには、サブスクリプション管理、ユーザー管理、接続管理、レポートツール統合、データ分析などが含まれ、IoTプラットフォームの機能を拡張します。
- GUI：エンドユーザー向けのインターフェースを提供するモジュールです。Firefox、Chromeなどの複数のブラウザ、およびモバイルアプリケーション（iOS、Android）をサポートします。
- データベース（Database）：デバイスからの情報やデータ、および設定情報を保存するデータベースです。

IoTプラットフォームは以下のシステムと連携します：

- **IPカメラCMSシステム**：画像によるポイント監視、IPカメラの管理、および録画、保存、再生業務を提供します。
- **IoTゲートウェイ**：センサーとゲートウェイ内のデバイス、およびポイント内の制御デバイスと連携するためにポイントに配置されるデバイスです。
- **IoTゲートウェイへの接続管理**：3G/4G/イーサネット/WiFiを通じてデバイスと接続を管理します。

IoTプラットフォームは以下のシステム統合プロトコルを使用します：

- IoTゲートウェイとIoTプラットフォーム間の統合プロトコル：MQTT。Thingsboardが定義したMQTTゲートウェイAPIに従います（詳細はこちらを参照）。
- IoTプラットフォームThingsboardとWebアプリ間の統合プロトコル：RESTFUL（HTTP）。Thingsboardが定義したAdministration REST APIに従います（詳細はこちらを参照）。
- WebアプリバックエンドとWebアプリフロントエンド間の統合プロトコル：RESTFUL（HTTP）。詳細なメッセージについては、AAAセクションを参照してください。
- WebアプリとCameraCMS間の統合プロトコル：RESTFULおよびiframe埋め込み。

IoTプラットフォームは以下の一般的なプログラミング言語を使用し、オブジェクト指向、モジュール化、最新のセキュリティを完全にサポートし、多くの拡張ライブラリを提供します：

- **Angular**：ユーザーインターフェースのプログラミング
- **Java**：IoTプラットフォームのバックエンドプログラミング
- **C#**：Webアプリのバックエンドプログラミング
- **Python**：IoTプラットフォームのSDKプログラミング

IoTプラットフォームはIPプロトコルで動作し、LAN/WANなどの現在の物理ネットワーク接続標準および3G/4G経由の無線接続と互換性があります。

ThingsBoard IoTソリューションの利点：

- 優れたパフォーマンス：単一のThingsBoardサーバーインスタンスは、20,000台以上のデバイスと1秒あたり30,000以上のMQTTメッセージのパブリッシュを継続的に処理でき、毎分約200万のメッセージを送信します。
- 信頼性の高いテレメトリデータの収集と保存：収集されたデータへのアクセスは、カスタマイズ可能なインターフェースの概要レポートウェブサイトを使用して行われます。
- マルチテナントのサポート：1つのテナントは、複数のゲートウェイ管理者、数百万のデバイス、および顧客を持つことができます。
- モジュラー設計：コアモジュール、IoTゲートウェイ統合モジュール、レポートモジュールなど。
- 直感的でわかりやすい管理インターフェース：管理インターフェースのカスタマイズはドラッグアンドドロップ操作だけで実行できます。
- データの視覚化：30以上の構成可能なウィジェットを即座に使用でき、統合されたエディターを使用して独自のウィジェットを作成する機能があります。ラインチャート、デジタルメーター、マップなどが組み込まれています。
- 水平スケーラビリティ：クラスター内の新しいダッシュボードサーバーが追加されると、サーバーリクエストとデバイスの数が線形に増加します。ダウンタイムやサーバーの再起動、アプリケーションの障害はありません。
- マイクロサービスとモノリシックの展開モデルのサポート：高可用性と水平スケーラビリティのためにマイクロサービスにアップグレードする機能を提供します。
- 多言語対応システム

2.2 IoTプラットフォームの運用モデル

2.2.1 全体的な運用モデル

ソフトウェアアーキテクチャ

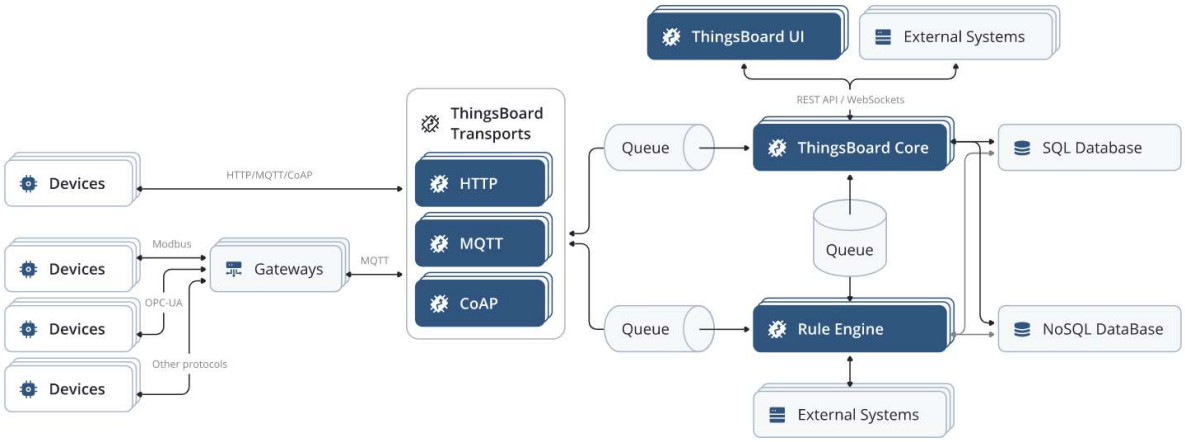

Figure 4: IoT Thingboardのソフトウェアアーキテクチャ

- Device：MQTTプロトコルを使用してIoTプラットフォームに接続するIoTデバイス
- Gateway：他のIoTデバイスを集約し、MTQQプロトコルを使用してIoTプラットフォームに接続するIoTデバイス
- Transport：各モジュールが異なる接続プロトコルを処理する独立したモジュールを含みます。Transportはデバイスからのパケットを受信し、解析し、キューに送信します。パケットがキューに正常に送信されて初めて、TransportはIoTデバイスにパケット受信を確認します。
- Queue：処理する必要があるパケットのキュー。モジュールは直接連結されず、キューを介して通信するため、簡単に拡張およびアップグレードできます。
- Core：デバイスの情報と接続状態を保存および管理します。CoreはRESTFULまたはWebsocketプロトコルを介してインターフェースレイヤーからの関数呼び出しも処理します。
- Rule Engine：キューからパケットを取り出し、定義済みのルールに従って処理および解析します。
- UI：ユーザーインターフェースを提供し、RESTFULまたはWebsocketプロトコルを介してCoreに接続します。
- データベース：SQLまたはNoSQLの2種類のデータベースをサポートします。
- External：外部システムがパケットの処理または表示に参加できます。

システム拡張時の展開モデル

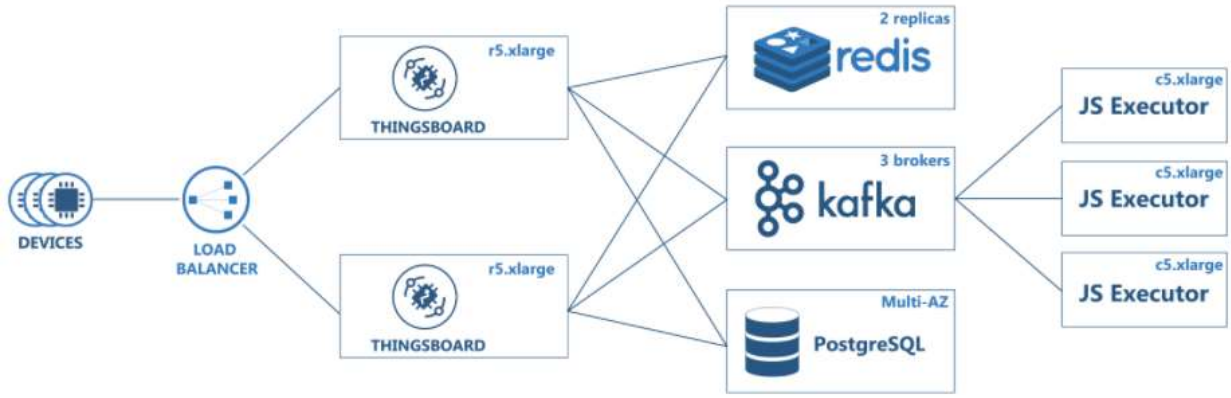

Figure 5: 拡張展開モデル

ThingsBoardは、数百万台のデバイスに対してスケーラブルな展開を実現するために、マイクロサービスアーキテクチャ（MSA）をサポートしています。MSAを展開することで、システム管理者は現在の負荷に応じてトランスポートサービス、ルールエンジン、Web-UI、およびJavaScriptエグゼキュータの数を柔軟に調整してクラスタを最適化できます。

MSAアーキテクチャモデルにおいて、私たちが提案するネットワークモデルも以下の標準モデルに従って展開されます：

提案されたネットワークおよびサーバーインフラ

- **バックエンドアプリケーションサーバークラスタ（Thingsboardコア）**：デバイスからのデータ受信とデータベースサーバーへのデータの書き込みおよび取得を処理するために使用されます。1+1台のサーバーがActive/Standbyモードで動作します。
- **データベースサーバークラスタ**：1+1台のサーバーがActive/Standbyモードで動作します。

提案されたネットワークセキュリティインフラ

- **ファイアウォールデバイスのクラスター（IPS統合）**：IoTゲートウェイからの接続を終了し、アクセス権を検証し、データをチェックして侵入攻撃や不適切なコンテンツを排除し、接続を背後のロードバランサーデバイスに転送します。

- **ロードバランサーデバイスのクラスター**：ゲートウェイから送信されるインターネット接続を受信し、その接続をThingsboardコアを実行しているバックエンドサーバーに分散します。

提案されたストレージインフラ

- **データベース全体**は、大容量ハードディスクを搭載し、RAID 5を実行するデータベースサーバークラスタに保存され、保存データの安全性を確保します。

2.2.2 管理および情報宣言ブロックの管理対象エンティティ

以下のエンティティを管理するためのAPIとプリセットインターフェースを提供します：

- システムユーザーとアクセス権限
- デバイス
- 階層化されたデバイスのメタデータ情報
- 警告情報（アラーム）
- ダッシュボード：パラメータを監視するための直感的なインターフェースをプリセットすることが可能
- デバイスからのパケット処理ルール

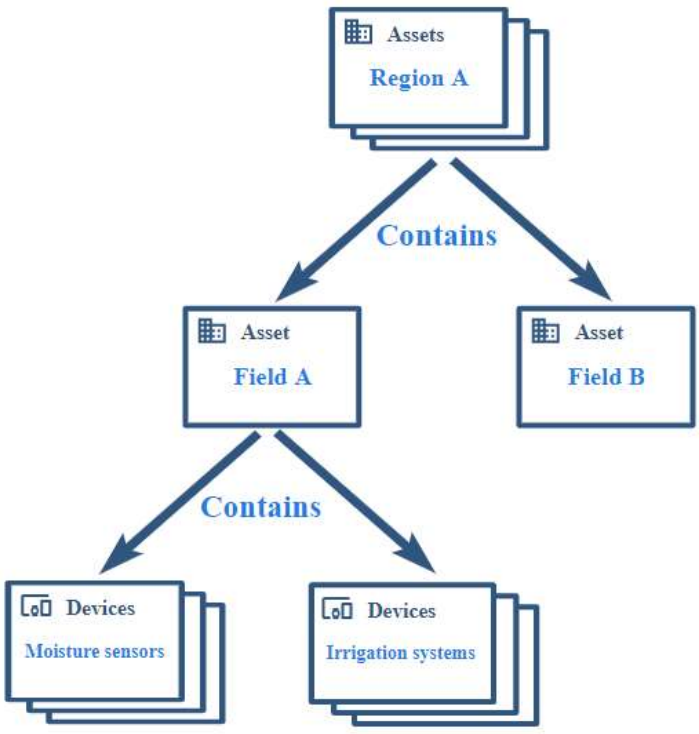

Figure 6: エンティティ管理API

2.2.3 デバイス属性の管理および宣言ブロック

デバイスの属性を管理するためのAPIとプリセットインターフェースを提供します：

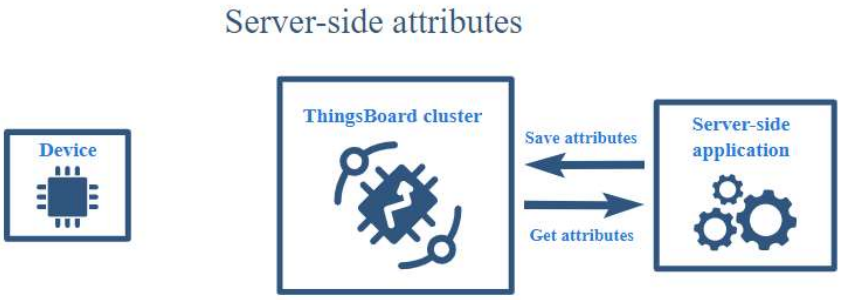

Figure 7: サーバー側属性

- サーバー側：IoTプラットフォームのアプリケーションによって管理される属性。IoTデバイス側からこれらの属性にはアクセスできません。

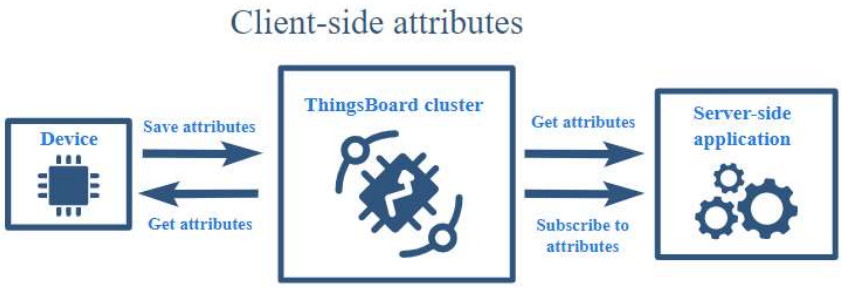

Figure 8: クライアント側属性

- クライアント側：ハードウェア/ソフトウェアバージョン、ハードウェアの詳細、デバイスの説明など、IoTデバイスのアプリケーションから管理および送信される属性。

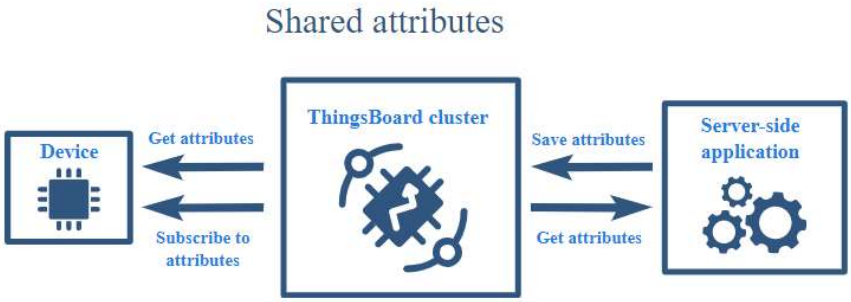

Figure 9: 共有属性

- 共有：IoTプラットフォームのアプリケーションによって管理され、IoTデバイス側からアクセスできる属性。

2.2.4 デバイスのテレメトリ管理ブロック

デバイスのデータを管理するためのAPIとプリセットインターフェースを提供します：

- リアルタイムまたは事前に設定された時間間隔のすべてのポイントからアップロードされるパケットを完全に受信および収集
- データを時間順に完全にデータベースに保存
- 時間パラメータまたは時間枠でクエリを実行できる
- データが更新された際に通知を受け取るための登録
- プリセットインターフェースコンポーネントによるデータの視覚化
- 定義済みのルールセットを使用してデータをフィルタリングおよび分析
- データが定義された閾値を超えたときに警告を実行
- デバイスデータを処理するために外部システムと接続

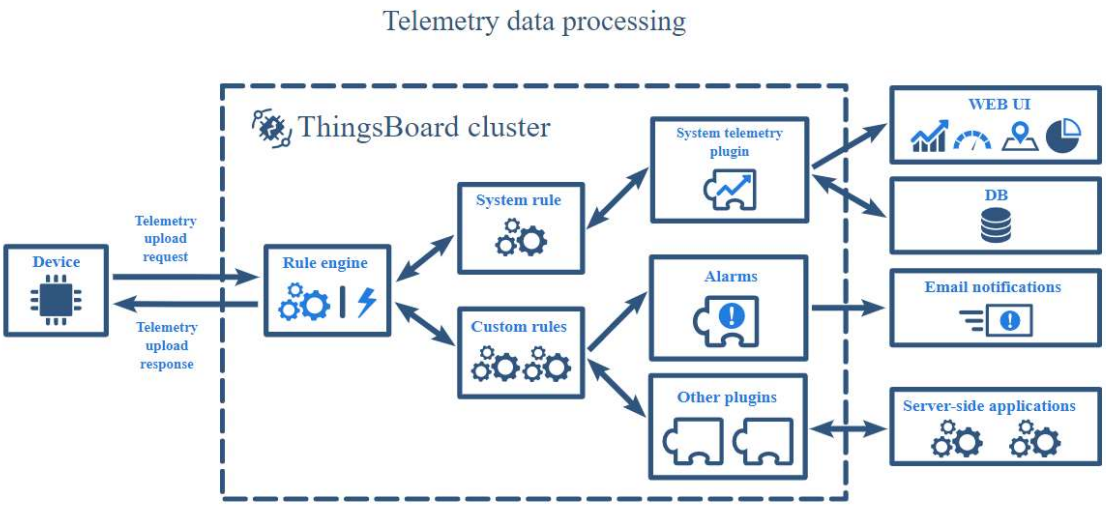

Figure 10: テレメトリデータ処理

2.2.5 パケット処理ルール管理ブロック

デバイスからのメッセージを処理するためのルールセットをユーザーが自分で作成できる直感的なツールを提供します。以下のルールは、IoTプラットフォームのソースコードを編集せずにインターフェース上で定義できます：

- メッセージのフィルタリングルール
- メッセージに情報を付加するルール
- メッセージの形式を変換するルール
- メッセージに影響を与えるルール
- メッセージ処理のためにサードパーティを呼び出すルール

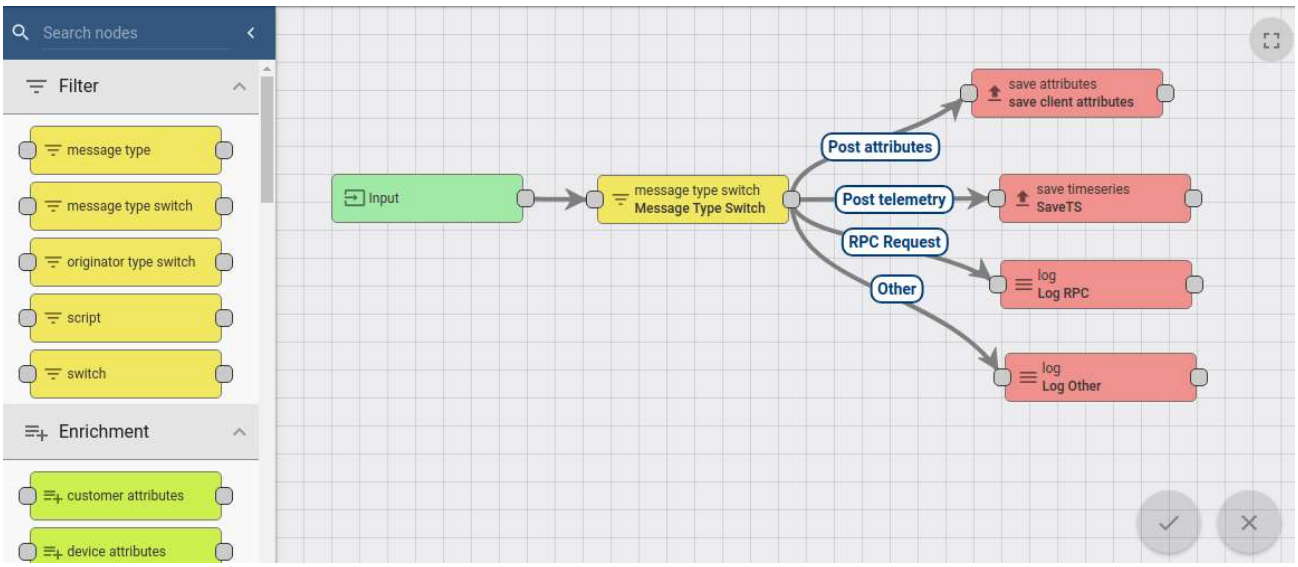

Figure 11: メッセージ処理ルールの設定

2.2.5.1 アラートブロック

デバイスからのデータに対してアラートを作成するためのAPIとプリセットインターフェースを提供します。

- 各アラームは削除および確認できます。
- アラームは、閾値を超えた場合にデバイスから起動されます。デフォルトでは、アラートはすべての関連エンティティ（親子関係）に伝達されます。アラームは開始時間、発信者、およびタイプによって識別されます。
- 次のアラートレベルがサポートされています：CRITICAL、MAJOR、MINOR、WARNING、INDETERMINATE。
- アラートとアラート生成ルール、およびインターフェース上のアラート閾値を構成できます。

2.2.5.2 デバイス接続状態管理ブロック

デバイスの接続状態を監視します。サポートされるイベントは次のとおりです：

- 接続 – デバイスが接続されたときにトリガーされます。
- 切断 – デバイスが切断されたときにトリガーされます。
- アクティブ – デバイスが属性を更新したり、RPCコマンドを受信したときにトリガーされます。
- 非アクティブ – デバイスが一定期間（5分、10分、30分など）アクティブでない場合にトリガーされます。

次のパラメータが保存されます：

- lastConnectTime – デバイスが最後に接続された時間を表します。
- lastDisconnectTime – デバイスが最後に切断された時間を表します。
- lastActivityTime – デバイスが最後に属性を更新したり、RPCコマンドを受信した時間を表します。
- inactivityAlarmTime – 最後に非アクティブイベントがトリガーされた時間を表します。

2.2.5.3 視覚化（UI）、データ分析およびレポートブロック

Thingsboard IoTプラットフォームは、以下の視覚化ウィジェットをサポートしてす：

デジタルメーター：温度、湿度、速度、およびその他の整数または浮動小数点値を表示するのに便利です。

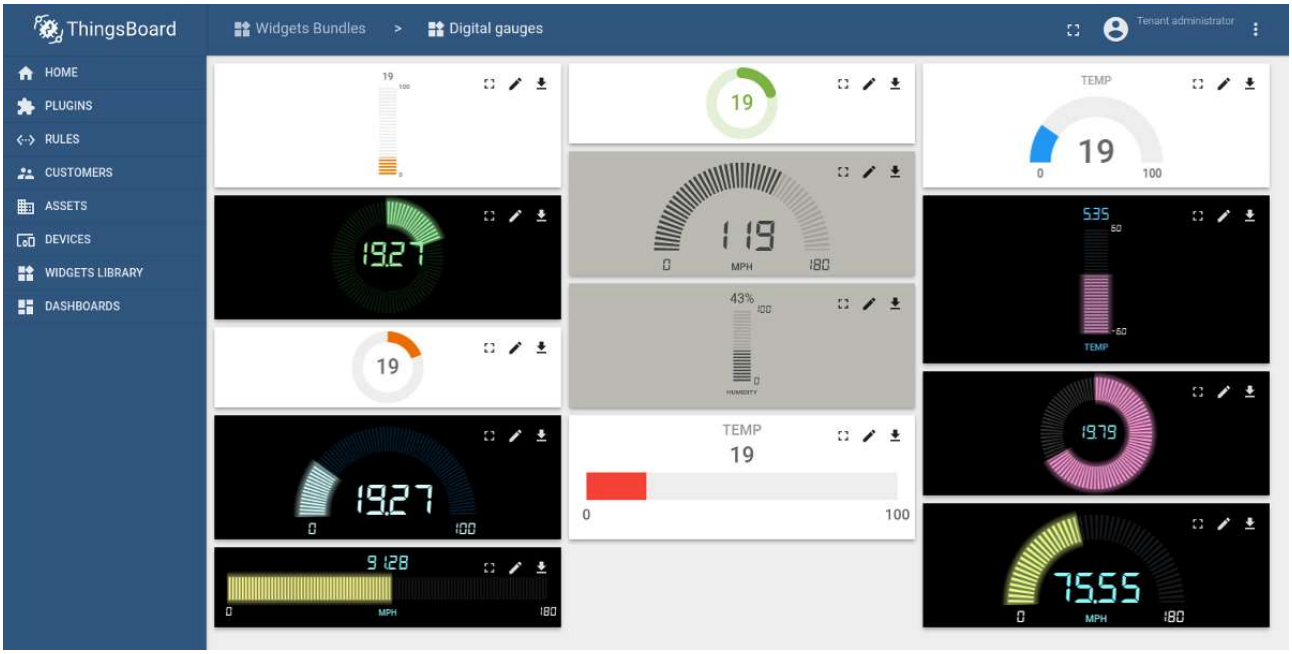

Figure 12: デジタルメーター：温度、湿度などを表示

グラフ：履歴データまたはリアルタイムデータを時間ウィンドウとともに視覚化するのに便利です。

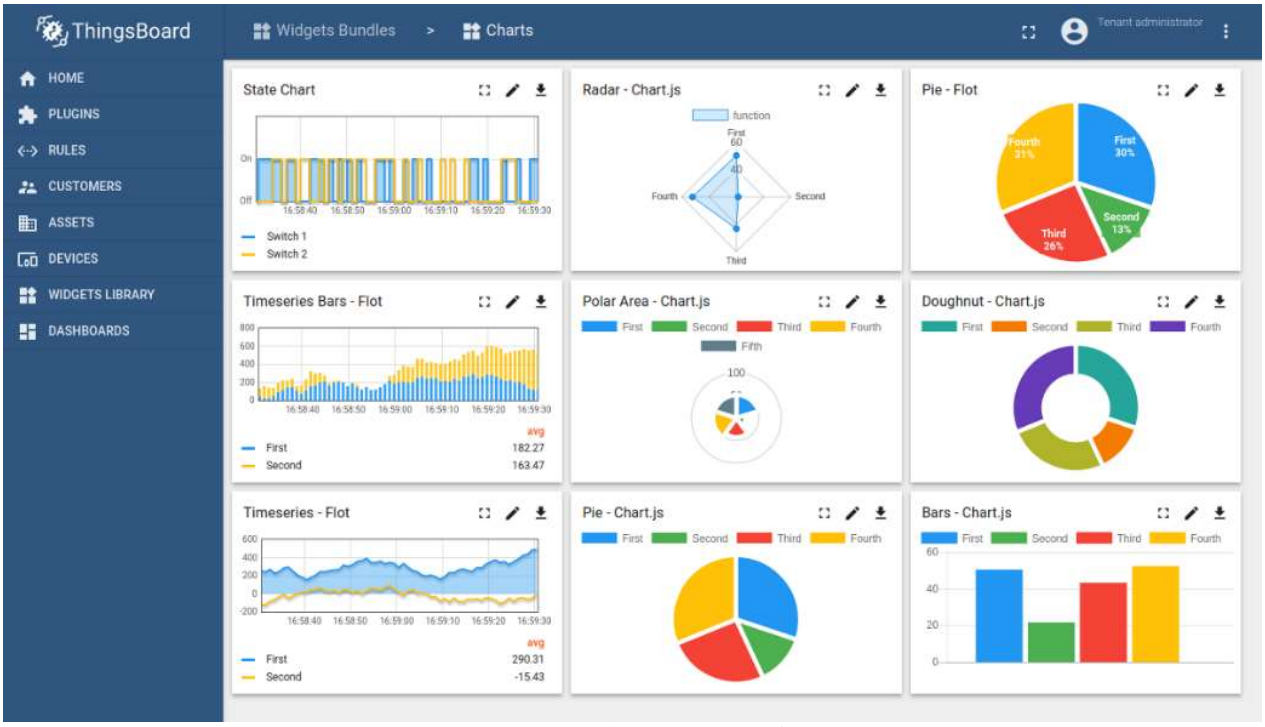

Figure 13: 履歴データの視覚化グラフ

マップ：デバイスの地理的位置を視覚化し、リアルタイムおよび履歴モードの両方でデバイスのルートを追跡するのに便利です。

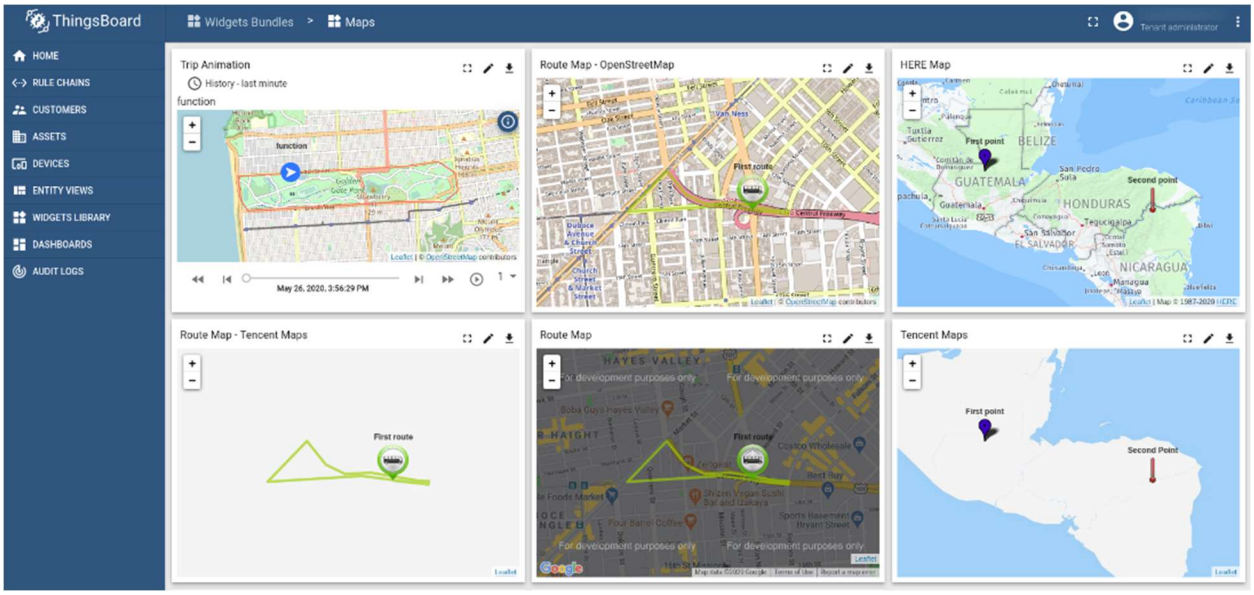

Figure 14: デバイスの地理的位置を視覚化するマップ

2.2.6 データベースブロック

PostgreSQLデータベースは、エンタープライズグレードのオープンソースデータ管理システムです。リレーショナルクエリ用のSQLと非リレーショナルクエリ用のJSONの両方をサポートしています。経験豊富な開発者のコミュニティによってサポートされており、非常に信頼性の高いDBMSシステムとなるために大きな貢献をしています。

PostgreSQLは、OracleやSQL　Serverなどの高価な商用データベースにしかない高度なデータ型とパフォーマンス最適化機能をサポートしています。

PostgreSQLは、多くの機能を提供し、開発者がアプリケーションを構築し、データの完全性を保護することによってフォールトトレラントな環境を構築するのに役立ちます。

PostgreSQLの最も優れた機能のいくつかは次のとおりです：

- さまざまなプラットフォームとの互換性：主要なすべての言語とミドルウェアを使用できます。
- 高度なロック機構の提供。
- 自動並列クエリのサポート。
- 多バージョン同時実行制御（MVCC）のサポート。
- 成熟したサーバーサイドプログラミング機能。
- SQL ANSI標準に準拠。
- クライアント-サーバーネットワークアーキテクチャの完全サポート。
- ログベースのレプリケーションとトリガーされたSSL。
- スタンバイサーバーと高可用性。
- オブジェクト指向とANSI-SQL2008互換。
- JSONのサポートにより、NoQueryなどの他のデータストアとリンクでき、ポリグロットデータベースのハブとして機能します。
- ハードウェアの能力に応じて、重要な情報に対してデュアルモジュールストレージモードで動作できます。

PostgreSQLの利点：これらの機能により、PostgreSQLは次のような優れた利点を提供します：

- PostgreSQLはLAMP環境で動的なウェブサイトやウェブアプリケーションを実行できます。
- PostgreSQLの事前書き込みログ（WAL）により、高い障害耐性を持つデータベースとなります。

- PostgreSQLのソースコードはオープンソースライセンスの下で無料で利用できます。これにより、ユーザーはビジネスのニーズに応じて自由に使用、修正、展開できます。
- PostgreSQLは地理空間オブジェクトをサポートしているため、位置情報サービスや地理情報システム（GIS）に使用できます。
- PostgreSQLは地理空間オブジェクトをサポートしているため、位置情報サービスや地理情報システム（GIS）のための地理空間データストアとして使用できます。
- 使いやすさ：直感的な管理ツールPgAdminは、GUIインターフェースでデータベースの管理を容易にします。
- システムの保守を最小限に抑える。
- 多くの冗長機能をサポート：ログ出荷、フェイルオーバー、ホットスタンバイ。

2.3 システムの可用性保証設計 (High Availability)

IoTプラットフォームThingsboardは、マイクロサービスアーキテクチャに基づく展開をサポートしており、すべてのサービスを1つのメインサーバー（アクティブ）と1つの待機サーバー（スタンバイ）で実行することで、システムの信頼性と高可用性を確保します。

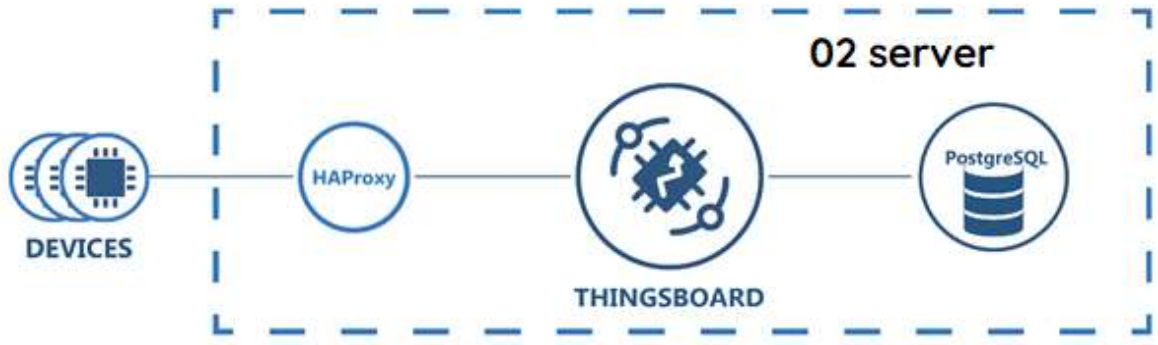

Figure 15: マイクロサービスの展開

メーカーのテスト結果および実際のテスト結果によると、モノリシックモデルでは3000～10000台のデバイス接続が可能であり、同時に500人のユーザーがアクセスしても、すべてのユーザーリクエストに100%応答することができます。

IoTプラットフォームは、接続デバイスの数が30,000から1,000,000に増加する場合、物理サーバーの数を増やすことで管理するデバイスの数を増やし、水平スケーリングをサポートします。

システムの水平スケーリングをサポートし、システムの可用性を年間99.95%に向上させるために、IoTプラットフォームはMicro Serviceアーキテクチャに基づいて設計およびインストールされています。これにより、システムのダウンタイムを月0.5時間未満に抑え、メンテナンスとトラブルシューティングの時間を最小限に抑えます。

- マイクロサービスは通常、一連の特定の機能や機能を実行し、ビジネスロジックと他のマイクロサービスとの通信を含みます。各マイクロサービスは、別々のハードウェアにインストールすることも、1つのマイクロサービスの複数のインスタンスを複数のハードウェアで同時に実行して負荷を分散することもできます。IoTプラットフォームのすべての機能は、個別のマイクロサービスに分割する必要があります。

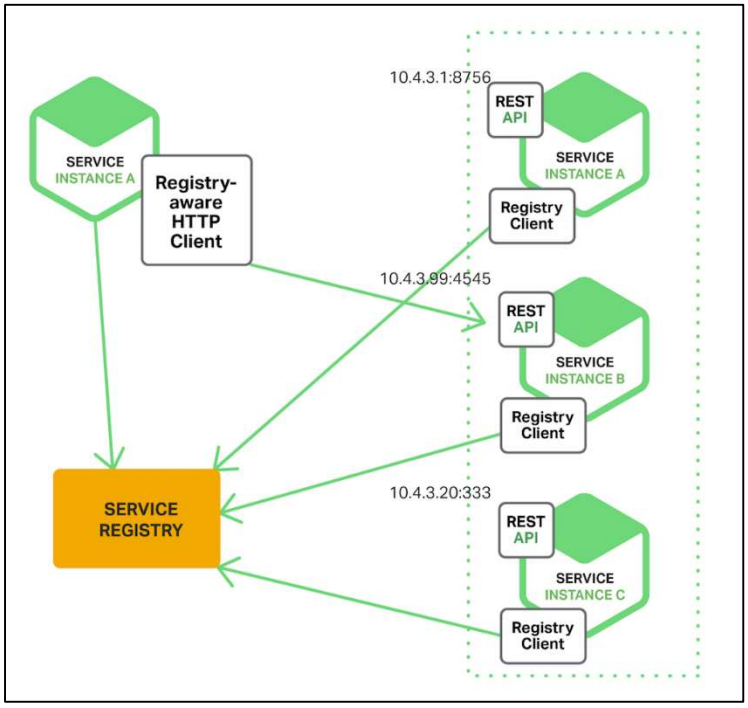

Figure 16: マイクロサービスの接続性

- マイクロサービスは直接接続されたり、密結合されるべきではありません。これにより、機能の変更や追加が難しくなります。代わりに、マイクロサービスはキュー（queue）を介して相互に接続する必要があります。これらのキューは、メッセージ数が増加したときの可用性、データの完全性、および拡張性を確保するように設計およびインストールされています。

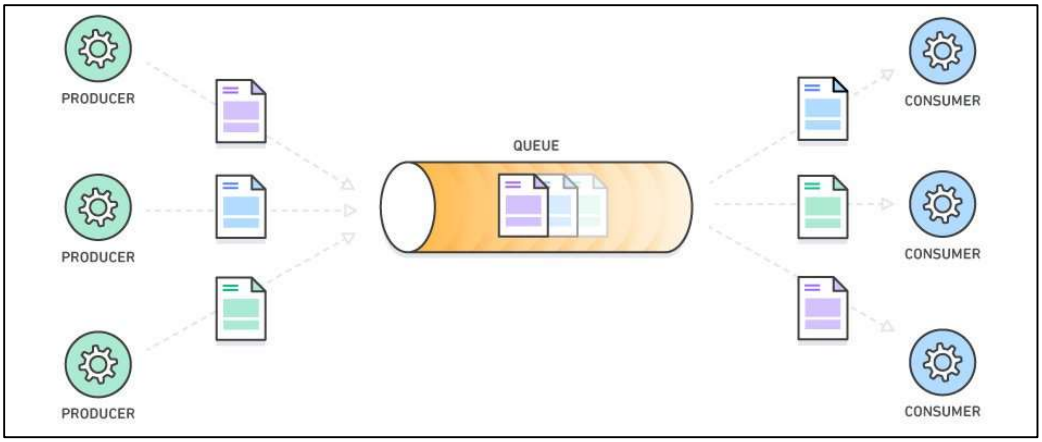

Figure 17: キューの仲介メカニズム

2.4 システムセキュリティ設計 (Security)

2.4.1 システムセキュリティ要件

- **コードのセキュリティ**
  - 入力情報の検証
  - 出力情報の暗号化
  - セッション管理
- **データベースのセキュリティ**
  - データベースの暗号化
- **ユーザーのセキュリティ**
  - アクセス制御
  - 通信保護
- **サーバーのセキュリティ**
- **ネットワークインフラのセキュリティ**

2.4.2 システムセキュリティソリューション

2.4.2.1 コードのセキュリティ

入力情報の検証:

- すべてのデータソースを特定し、それらを信頼できるものと信頼できないものに分類します。信頼できないソースからのすべてのデータ（例：データベース、ファイルストリームなど）を検証します。
- アプリケーションの集中入力検証ルーチンを確立します。
- すべての入力ソースに適切な文字セット（例：UTF-8）を使用します。
- 検証前にデータを一般的な文字セットにエンコードします（正規化）。
- すべての検証エラーは入力拒否につながります。
- システムが拡張UTF-8文字セットをサポートしているかどうかを確認し、サポートしている場合はUTF-8デコードが完了した後に検証します。
- すべてのパラメータ、URL、およびHTTPヘッダーの内容（例：クッキーの名前と値）を含む、処理前にクライアント提供のすべてのデータを検証します。入力検証には、JavaScript、Flash、またはその他の埋め込みコードからの自動投稿が含まれます。

出力情報の暗号化

- 信頼できるシステムでのみ暗号化を実行する
- 各種の出力暗号化に対して標準チェックルーチンを確立する
- クライアントマシンに返されるすべてのデータを暗号化する
- 予定されたインタープリターで安全とされる文字以外はすべて暗号化する
- SQL、XML、およびLDAPのクエリに対して不正なデータ出力をすべて除去する
- OSコマンドに対して不正なデータ出力をすべて除去する

セッション管理

- サーバーセッション管理またはアプリケーションフレームワークのコントロールを使用する。アプリケーションはこれらのセッション識別子のみを有効と認識するべきである
- セッション識別子の生成は常に信頼できるシステムで実行する必要がある
- セッション管理コントロールは、十分にランダムなセッション識別子を確保するために適切なアルゴリズムを使用するべきである
- セッション識別子を含むクッキーのドメインとパスを、ウェブサイトに適した制限値に設定する
- グアウト機能はセッションまたは関連する接続を完全に終了する

2.4.2.2 データベースのセキュリティ

- パラメータ化されたクエリの使用：クエリコマンドとデータを分離する

- 入力検証と出力エンコードを使用し、メタ文字を適切に処理する。失敗した場合、データベースコマンドを実行しない
- 変数がパラメータ化されていることを確認する
- アプリケーションは、データベースにアクセスする際に可能な限り最低特権を使用するべきである
- データベースアクセスに安全な資格情報を使用する
- 接続文字列をアプリケーションにハードコーディングしない。接続文字列は信頼できるシステム上の別個の設定ファイルに保存し、暗号化する必要がある
- データアクセスにはストアドプロシージャを使用し、データベース内の基本テーブルへの権限を取り除くことを許可する

2.4.2.3 ユーザーのセキュリティ

アクセス制御:

- 信頼できるシステムオブジェクトのみを使用してアクセス権の決定を行います。例えば、サーバーサイドセッションオブジェクトなどです。
- アクセス権を検査するためにWebコンポーネントを使用します。
- アクセス制御が失敗した場合、安全に失敗させます。
- アプリケーションがセキュリティ構成情報にアクセスできない場合、すべてのアクセスを拒否します。
- サーバーサイドスクリプトによって生成された要求やAJAXやFlashなどのクライアントサイド技術からの要求を含むすべての要求に対してアクセス権の制御を実施します。
- 特権ロジックを他のアプリケーションコードから分離します。

通信チャネルの暗号化:

- すべての機密情報の伝送には暗号化を実施します。TLSを使用して接続を保護し、必要に応じてファイル暗号化やHTTP以外の接続の暗号化を追加します。
- TLS証明書は有効であり、正しいドメイン名を持ち、有効期限が切れておらず、中間証明書が必要な場合にはインストールされている必要があります。
- TLS接続が失敗した場合、不安全な接続にフォールバックしてはなりません。
- 認証されたアクセス権を要求するすべてのコンテンツおよびその他のすべての機密情報に対してTLS接続を使用します。

2.4.2.4 サーバーのセキュリティ

- 最新のセキュリティパッチを適用する
- Kaspersky Server Enterprise 2023のアンチウイルスをインストールする

2.4.2.5 ネットワークインフラのセキュリティ

- Firewall Checkpoint 4200を使用してアクセス制御機能を展開する
- Juniper IPS XG2000を使用して侵入防止機能を展開する

2.4.3 セキュリティ評価の方法

- tslintツールを使用してOWASPガイドラインに従ったセキュリティルールを構成する
- Acunetixツールを使用してセキュリティスキャンを実行する
- 展開後のシステムには、重大なセキュリティ脆弱性（OWASPトップ10）は残らない

2.5 システム拡張性設計 (Scalability)

IoTプラットフォームの開発者は、任意の数のデバイス、テレメトリ送信サイクル、およびパフォーマンステストシナリオをシミュレートすることによってシステムパフォーマンスを評価するためのガイドラインとツールを提供しています。詳細はこちらを参照してください。メーカーが発表したテスト結果は、以下のように線形パフォーマンスを示しています：

Instance Type	Instance details	Database Type	Device API	Number of devices	Delay between messages	Maximum number of messages
t2.micro	1 vCPUs for a 2h 24m burst, 1GB	PostgreSQL	MQTT	500	1000 ms	~450/sec
t2.medium	2 vCPUs for a 4h 48m burst, 4GB	PostgreSQL	MQTT	900	1000 ms	~780/sec
c5.large	2 vCPUs , 4GB	PostgreSQL	MQTT	1100	1000 ms	~1020/sec
t2.xlarge	4 vCPUs for a 5h 24m burst, 16GB	PostgreSQL	MQTT	1800	1000 ms	~1700/sec
t2.xlarge	4 vCPUs for a 5h 24m burst, 16GB	Cassandra	MQTT	3000	1000 ms	~3000/sec
m5.xlarge	4 vCPUs, 16GB, 150GB SSD mounted	Cassandra	MQTT	3500	1000 ms	~3500/sec
m5.xlarge	4 vCPUs, 16GB, 150GB SSD mounted	Cassandra	HTTP	2000	1000 ms	~950/sec

Figure 18: システム性能テストの結果

メーカーの理論に基づき、以下のようにサーバーハードウェアの総構成を決定します：

- 1,000ポイント: 2 CPU、4 GB RAM
- 10,000ポイント: 20 CPU、40 GB RAM
- 30,000ポイント: 60 CPU、120 GB RAM

私たちは、メーカーのガイドラインに従って、モノリシックおよびマイクロサービスのシナリオをシミュレートし、1,000、10,000、30,000ポイント（養殖場やエビ池）で負荷

性能テストを実施しました。これには10,000テレメトリ/秒の設定が含まれており、各テレメトリは200バイトのJSON形式のデータです。このシナリオは、実際のMobifone Globalの使用ケースよりも10〜20倍高い負荷をかけて負荷性能テストを行い、メーカーの発表とテスト結果に一致しました。

使用したハードウェアは、Ciscoサーバーの Intel CPU i5-9400、16 GB RAMです。

私たちのテスト結果は、CPUとRAMが許容範囲内（CPU：60〜70%、RAM：50%未満）であることを示しました。平均応答時間は1〜3秒、ページの完全な情報表示時間は5秒未満でした。

測定方法:

- 使用ツール: Gatling MQTT Plugin Fork
- IoTゲートウェイからプラットフォームへのメッセージ送信モデル:

- ハードウェア: Ciscoサーバー、Intel CPU i5-9400、RAM 16GB
- シミュレーションシナリオの例:

```
class MqttSimulation extends Simulation {
    val mqttConfiguration = mqtt
        // MQTT broker
        .host("tcp://localhost:1883")

    val connect = exec(mqtt("connect")
        .connect())

    // send 100 publish MQTT messages
    val publish = repeat(100) {
        exec(mqtt("publish")
            // topic: "foo"
            // payload: "Hello"
            // QoS: AT_LEAST_ONCE (1)
            // retain: false
            .publish("foo", "Hello", QoS.AT_LEAST_ONCE, retain =
false))
            // 1 seconds pause between sending messages
            .pause(1000 milliseconds)
```

```
        }

    val disconnect = exec(mqtt("disconnect")
      .disconnect())

    val scn = scenario("MQTT Test")
      .exec(connect, publish, disconnect)

    setUp(scn
        // linearly connect 10 devices over 1 second
        // and send 100 publish messages
        .inject(rampUsers(10) over (1 seconds))
    ).protocols(mqttConfiguration)
}
```

2.5.1 1,000ポイントに対するCPUおよびRAMの測定結果

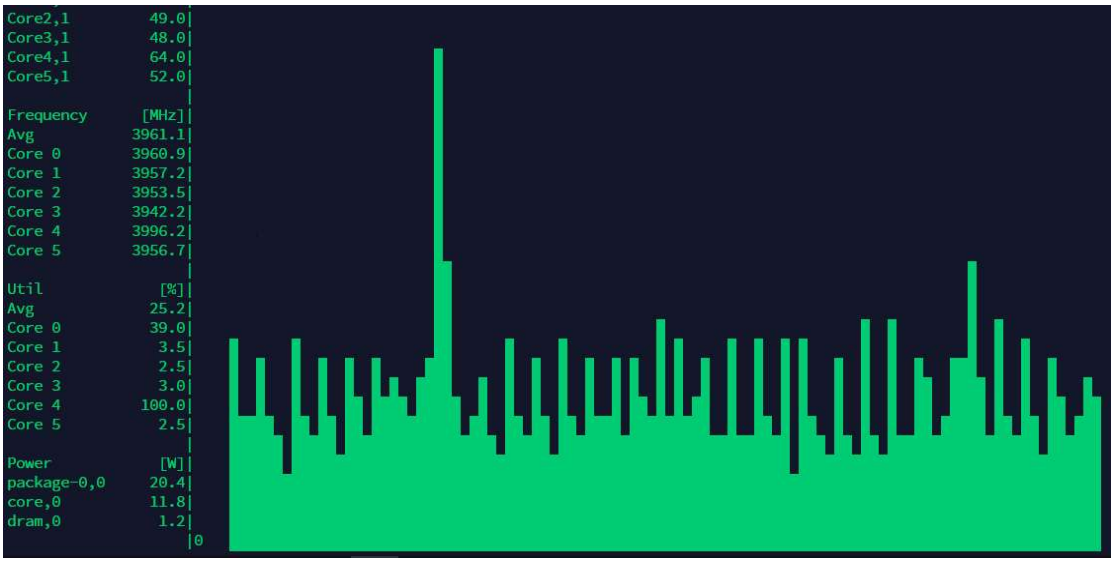

Figure 19: ウェブサーバーのCPU使用率グラフ

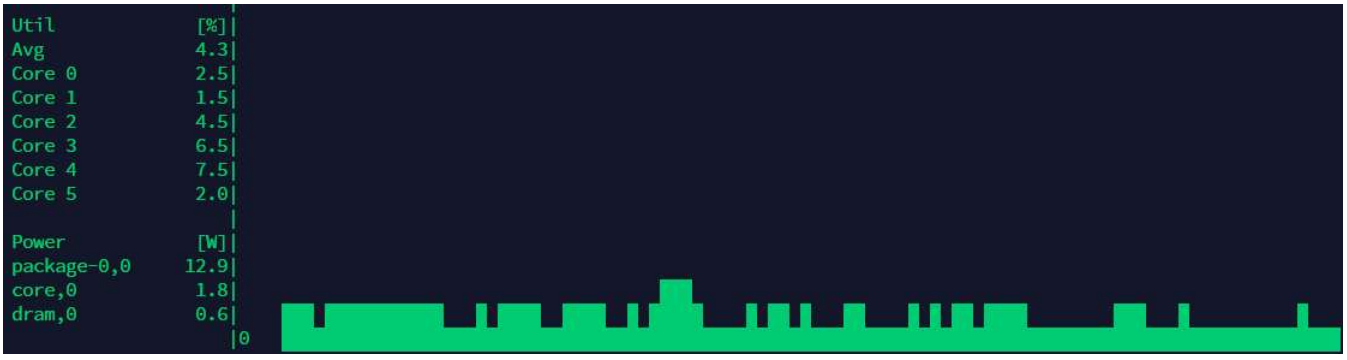

Figure 20: データベースサーバーのCPU使用率グラフ

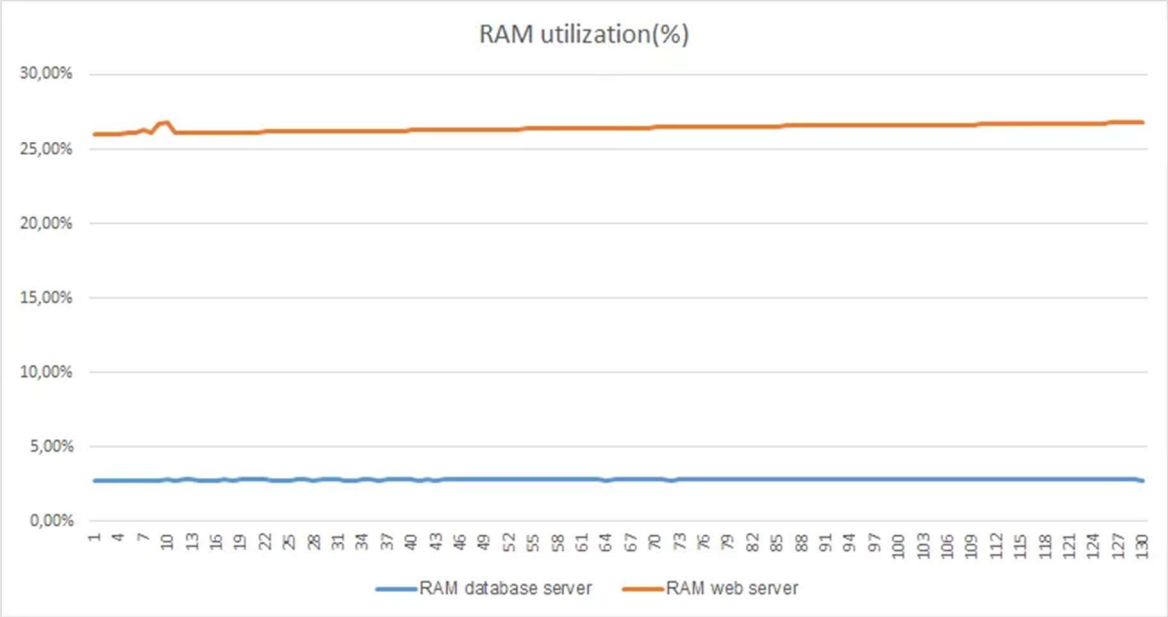

Figure 21: ウェブサーバーおよびデータベースサーバーのRAM使用率グラフ

2.5.2 5,000ポイントに対するCPUおよびRAMの測定結果

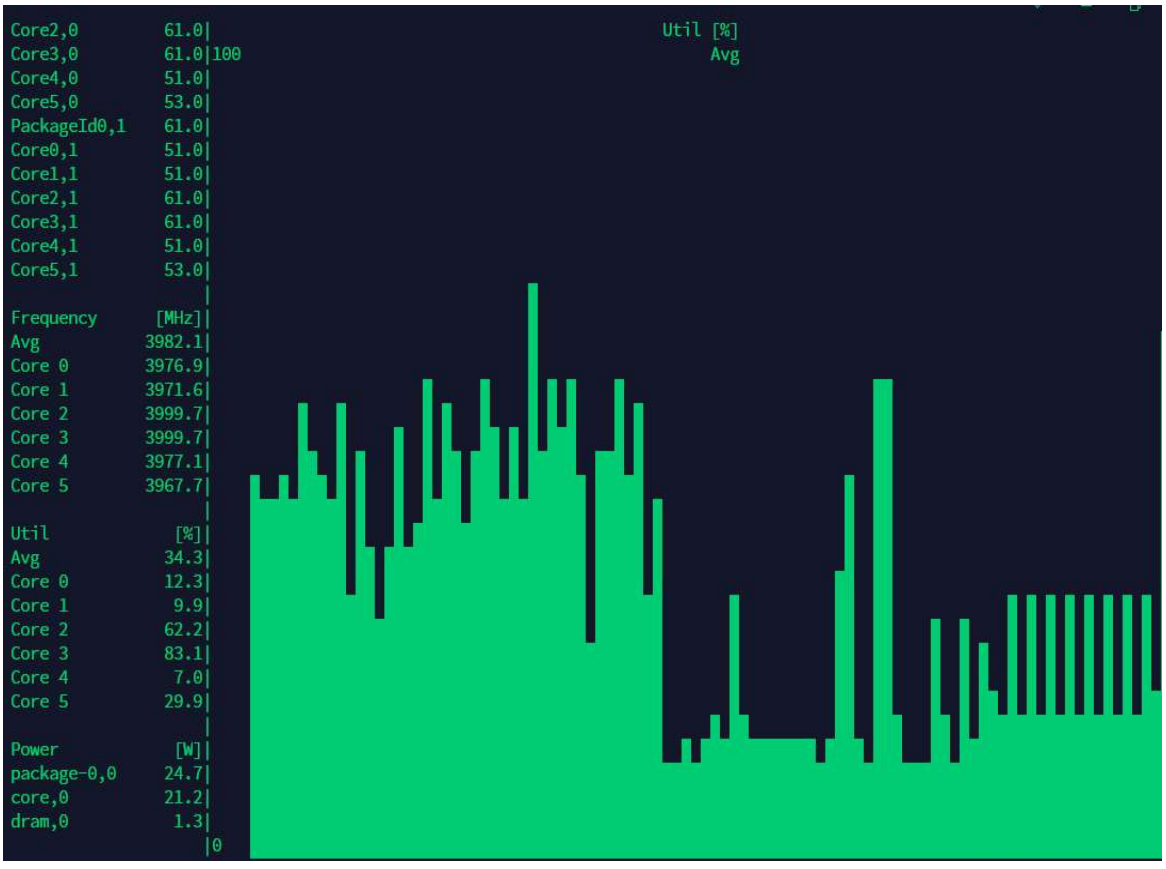

Figure 22: ウェブサーバーのCPU使用率グラフ

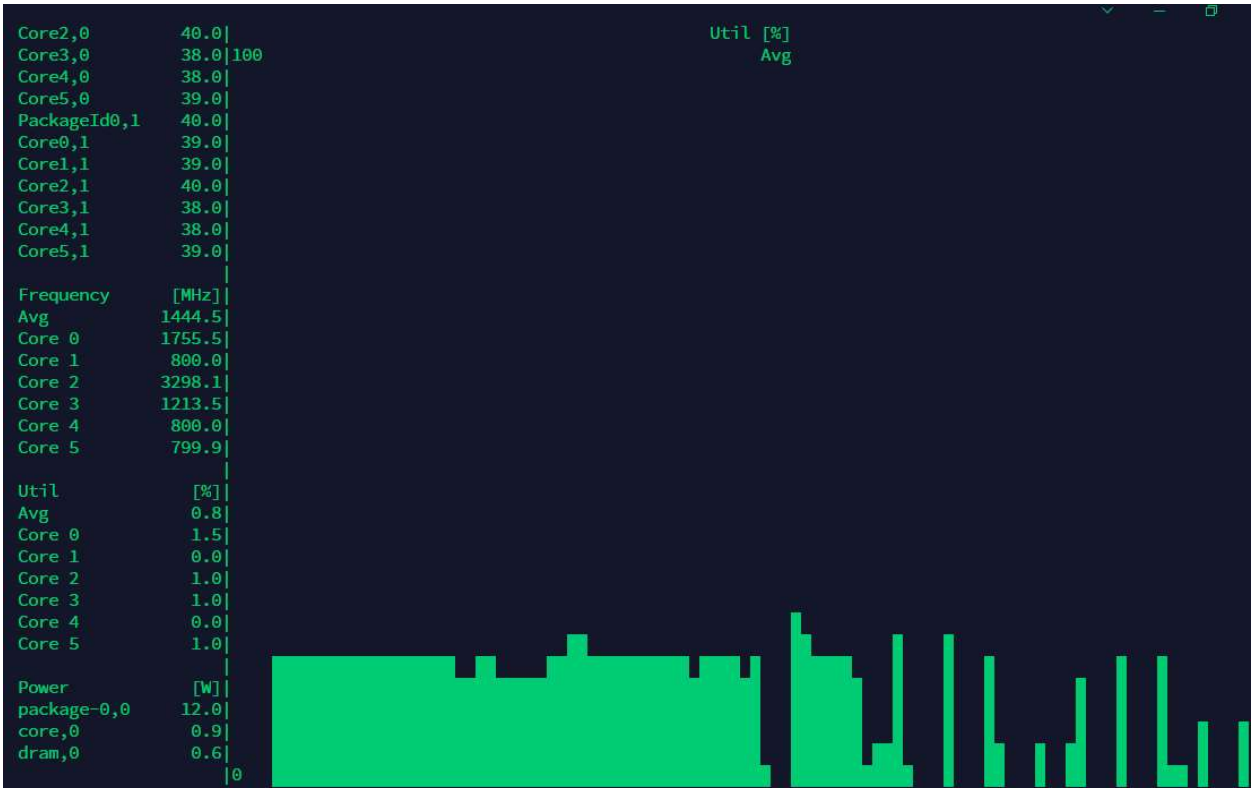

Figure 23: データベースサーバーのCPU使用率グラフ

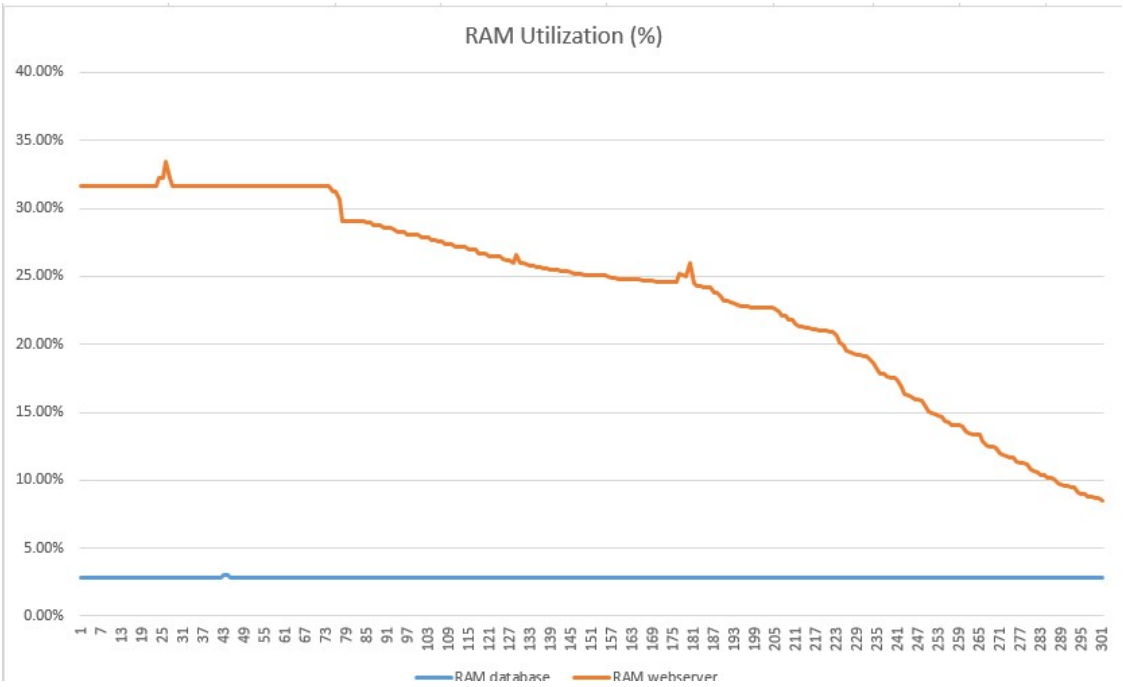

Figure 24: ウェブサーバーおよびデータベースサーバーのRAM使用率グラフ

2.5.3 10,000ポイントに対するCPUおよびRAMの測定結果

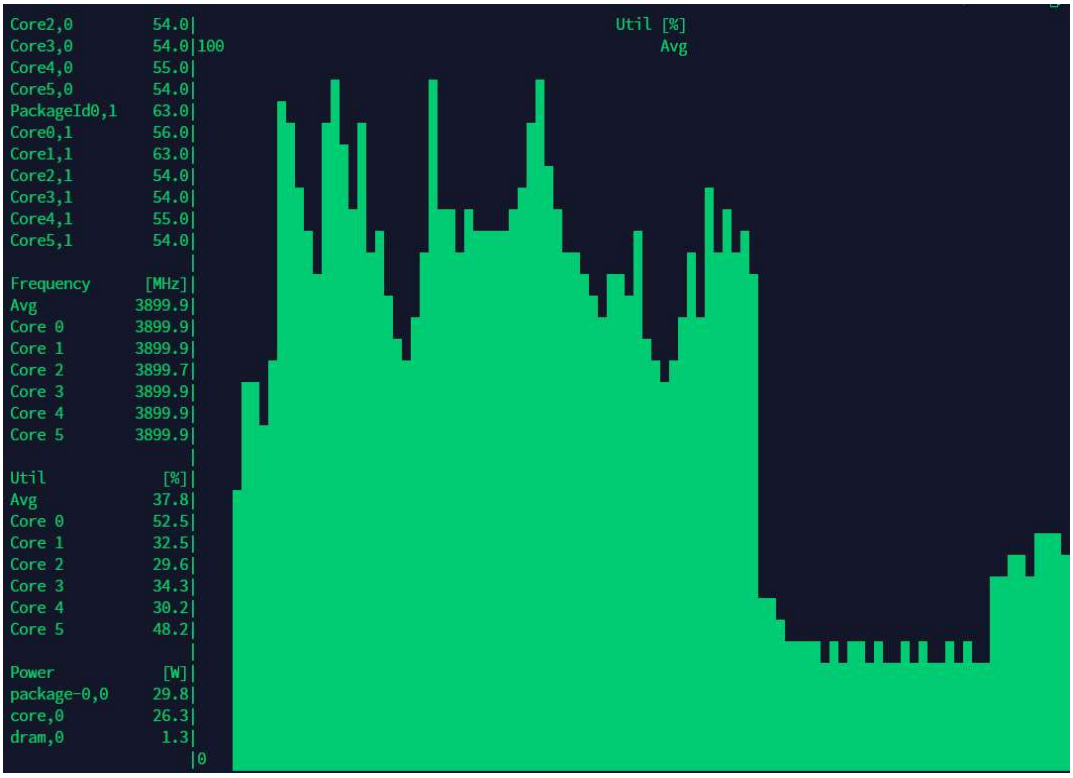

Figure 25: ウェブサーバーのCPU使用率グラフ

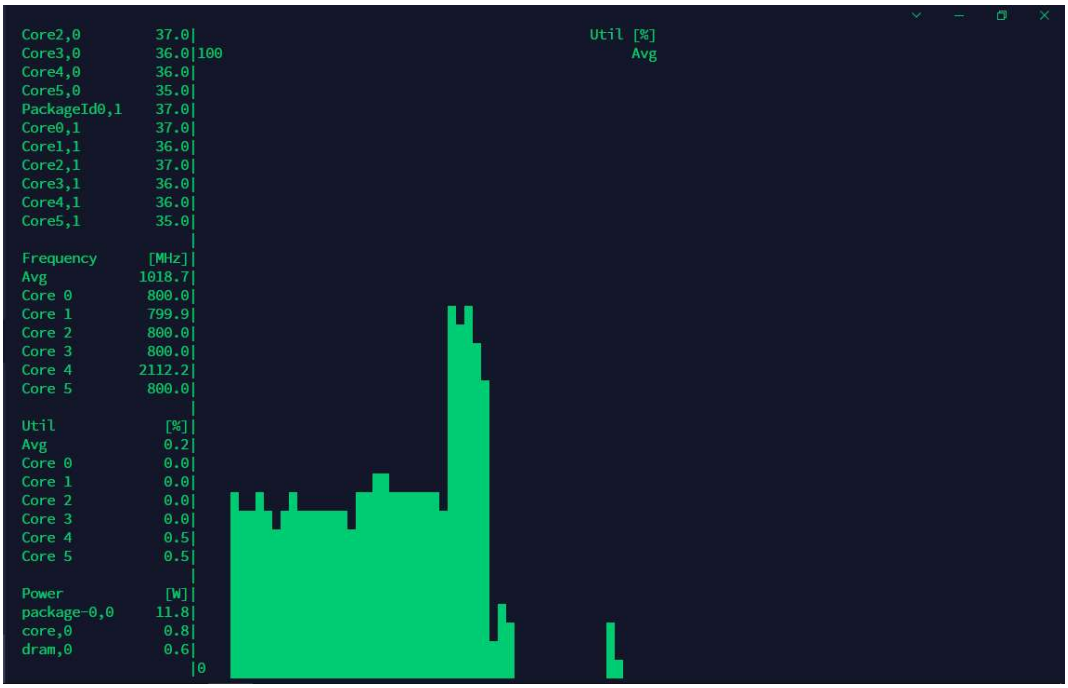

Figure 26: データベースサーバーのCPU使用率グラフ

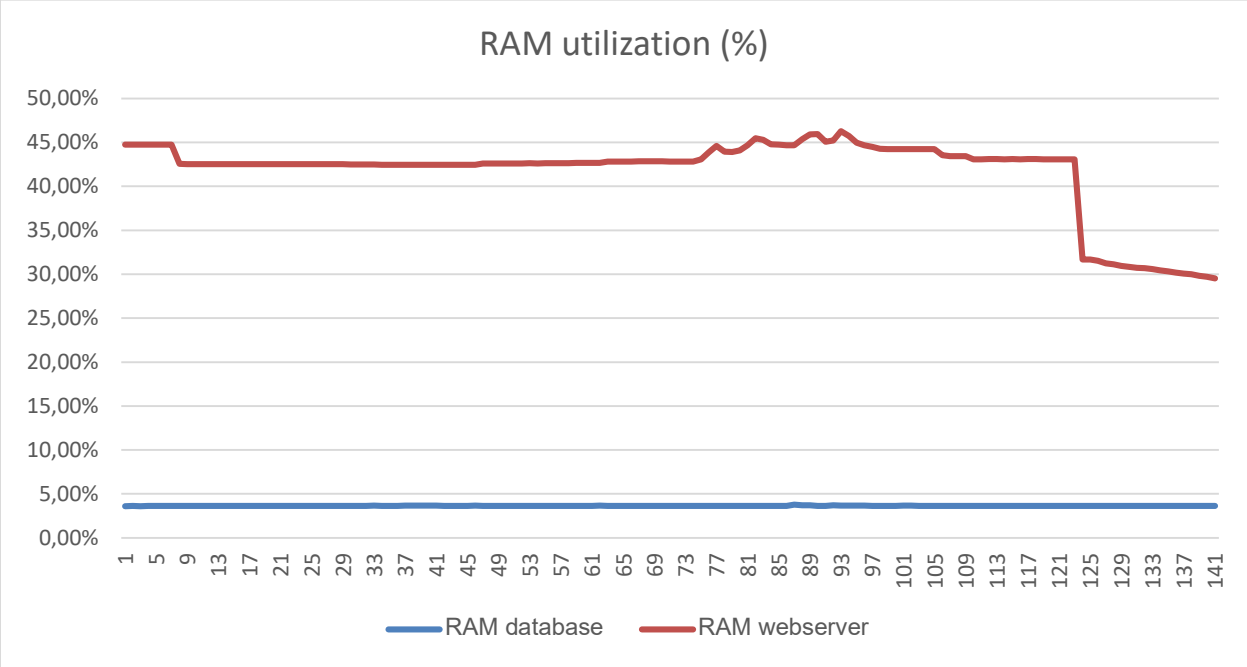

Figure 27: ウェブサーバーおよびデータベースサーバーのRAM使用率グラフ

2.5.4 サービスを実行するサーバーの設計

メーカーの推奨事項と実際のテスト結果に基づき、以下のようにサーバーを投資することをお勧めします

サービスの種類	サーバーの数	CPUコアの総数	RAMの総数 (GB)	HDDの総数 (TB, 最初の1年間のサービス運用)
IoT Platform	2	32	64	4TB

2.6 システムのバックアップと復元 (Back-up and Recovery)

次の戦略を適用する：

- フルバックアップ
- 増分バックアップ

バックアップするデータ：

- データベース
- ログ
- 設定ファイル
- データファイル
- ソースコード

バックアップメディアのサポート：

- ディスク
- テープ
- ブロックストレージ
- NAS
- SAN

バックアップ方法：

- データベース: データベースが提供するツールを使用
- ログ: bashスクリプトとrsync
- 設定ファイル: bashスクリプトとrsync
- データファイル: bashスクリプトとrsync
- ソースコード: git

バックアップ周期：バックアッププロセスを自動化するためにcronjobまたはスケジューラを使用し、バックアッププロセスに問題が発生した場合にメールを送信するように設定します。

- データベース: 週次、日次、時次
- ログ: 週次
- 設定ファイル: 日次
- データファイル: 日次

- ソースコード: バージョンアップごと

古いバックアップファイルを削除（保持）するために、Linux上でrsyncとbashスクリプトを使用してバックアップファイルの保持期間（日数）を設定します。

特にデータベースについては、データベースが提供するツールを使用して、以下のスケジュールでバックアップを実施します：

- フルバックアップ：週次
- 差分バックアップ：日次
- 増分バックアップ：時次

上記の戦略に基づくバックアップから、復元が必要なデータの種類に応じて、1時間から3時間の間にシステムを最短で復元できます。rsyncなどのツールは、バックアップおよび復元プロセスにおいて詳細で完全なエラーログを出力する能力を持ち、処理の監視とデバッグを支援します。

2.7 システム運用監視

2.7.1 Thingboardのシステムログ

Thingboard IoTプラットフォームのログは、次のディレクトリに保存されます：
/var/log/thingsboard

詳細なエラーとその解決方法を確認するためには、次のコマンドを使用してエラーログをチェックし、デバッグします：

cat /var/log/thingsboard/thingsboard.log | grep ERROR

2.7.2 Linuxオペレーティングシステムのシステムログ

Linuxオペレーティングシステムのログは通常:

/var/log/message

トラブルシューティングやデバッグを行うためには、以下のコマンドを使用してエラーの詳細やその修正方法を確認できます。

cat /var/log/message | grep error

2.7.3 LinuxシステムでのNagiosによるリソース消費の監視と警告

Nagiosは、企業がネットワークインフラストラクチャやサービスを監視し、問題を特定して解決するための非常に強力な無料ネットワーク監視およびサービス監視システムです。

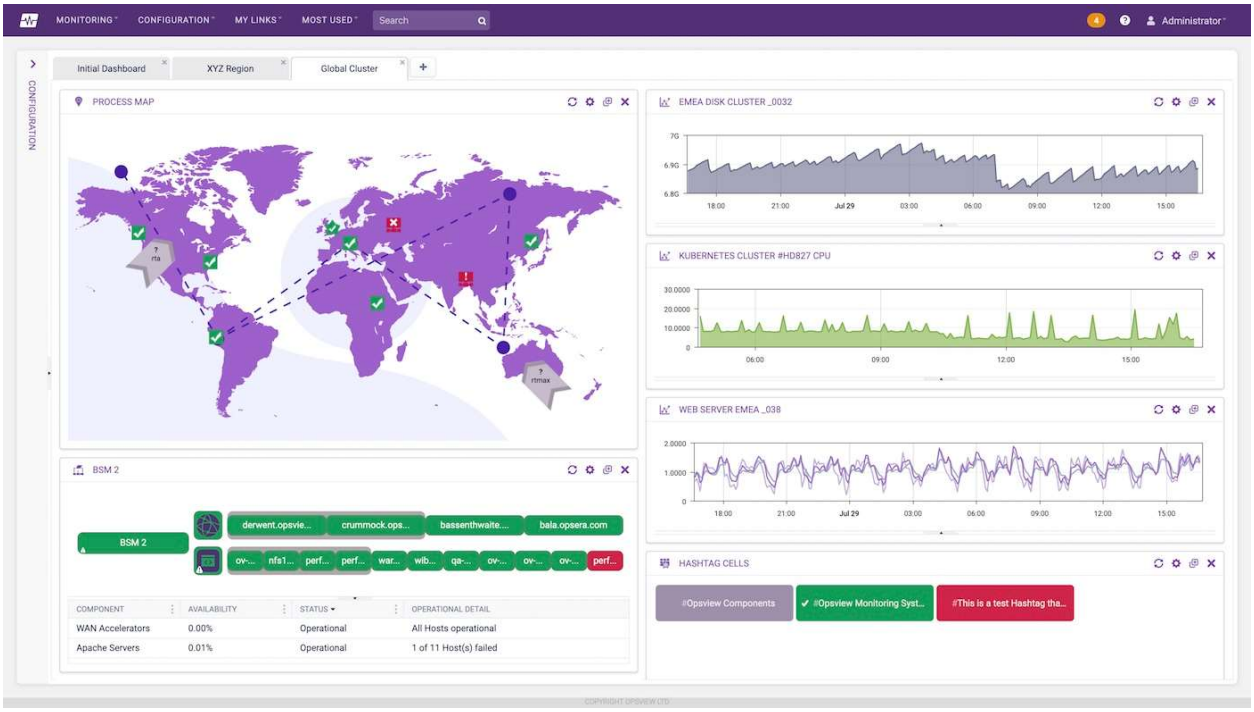

Figure 28: Nagiosによる監視画面

Nagiosは以下のようなサービスを提供する能力を持っています:

- サーバーのリソース監視（プロセッサーロード、使用済みディスク容量、ログなど）。多くのネットワークオペレーティングシステム、例えばMicrosoft Windowsは、エージェントモニターを使用します。
- ハードウェアの監視（温度、アラームなど）、収集されたデータを特定のプラグインにネットワーク経由で送信する能力。
- Nagios Remote Plugin Executorを使用したリモートモニタリング、またはSSHや暗号化されたSSLトンネルを介したリモート監視。並行サービスチェックと自動ログローテーションをサポート。
- バックアップ監視サーバーの展開をサポートし、パフォーマンスデータとデータベース補助をチャート化する。Webインターフェースを介して現在のネットワークステータス、通知、障害履歴、ログファイルなどを表示する。