IoT基盤サービスHABANEROTSのマルチテナント化と顧客志向のリソース管理

HABANEROTS IoT Platform Service Incorporating Multitenancy and Customer-Oriented Resource Management

樽家　昌也 TARUI Masaya　　望月　翔平 MOCHIZUKI Shohei

東芝が開発しているHABANEROTSは，IoT（Internet of Things）データ収集とアセット管理のプラットフォームであり，テナント（利用者や顧客）が，直接特有の開発に専念できるように設計されていて，アプリケーション運用のための多くの機能を担っている。現在，複数のテナントがHABANEROTSを利用しているが，より多くのテナントへ継続的にサービスを提供するためには，その運用コストを削減する必要があった。

これを解決するため，HABANEROTSにリソースをシェアするマルチテナント環境を構築した。リソースの特徴を考慮するのに加え，ホスティング環境に対して適切な設定とポリシー制御による設定制約を導入することでセルフサービス化も行い，更なる運用コスト削減と柔軟なサービス設定の実現を達成した。

Toshiba Corporation is developing HABANEROTS, a platform for Internet of Things (IoT) data collection and asset management services to allow tenants, such as users and customers, not only to directly concentrate on business development, but also to provide varied functions to operate their applications. Having HABANEROTS be used by multiple tenants is an important part of reducing operational costs so that services can be offered continuously to more tenants.

To address this issue, we have constructed a multitenant environment on HABANEROTS to share its resources while taking into consideration the characteristics of each resource, as well as providing a self-hosting service by introducing appropriate settings and policy controls to hosting environments, thereby reducing operational costs and providing flexible service settings.

1. まえがき

東芝は，CPS（サイバーフィジカルシステム）の共通機能を提供するプラットフォームとして，HABANEROTSを構築し，CPS開発運用基盤の統合による開発運用コストの削減とサービスビジネス競争力の強化を目指している[(1)–(3)]。顧客は，HABANEROTSの利用により，IoT機器データを活用する新たなCPSサービスを容易に開始できる。一方，継続的なサービス提供には運用コストの削減が必要であり，特に導入を容易にするために立ち上げ時のコストを抑える施策を検討していた。

そこで，同じインスタンスで複数のビジネスを扱うマルチテナント化を導入し，HABANEROTSの各リソースに適した管理方法で，ビジネス間のデータの分離を実施した。更に，顧客のアプリケーションを動作させるホスティング環境においても，これまで顧客ごとに手動でカスタマイズしていた部分にポリシーベースの管理を導入し，顧客自身が行えることを増やした。これにより，細やかなサービス設定を迅速に行えるようにするとともに，管理コストの削減を図った。

ここでは，これらのマルチテナント化とホスティング環境のポリシー制御導入について述べる。

2. マルチテナント化

マルチテナント化は，一般に複数のテナントが一つのシステムやプラットフォームを共有して利用することを指す。つまり，従来それぞれのインスタンス上で動いていたサービスを，同じインスタンス上で動かしながらもデータとしては独立していて，ほかのテナントとは隔離された状態を保つものである。これによって，システムの運用コストを削減し，リソースの効率的な利用が可能となる。また，テナントごとに独立した環境で相互にデータが参照されることがないため，情報の保護や，不正アクセス防止，個人情報の適切な取り扱いといったセキュリティーやプライバシーの要件も満たせる。

HABANEROTSは，マイクロサービスアーキテクチャーを採用し，その管理にオーケストレーションツールであるKubernetes®[(4)]を用い，複数のテナント間で共通してKubernetesを利用するマルチテナント環境を構築した（**図1**）。

従来は，顧客ごとに名前空間（Namespace）を切り，ポッド（Pod）群を用意してサービスを提供していた。しかし，今回のアプローチでは，顧客からのリクエストがKubernetesに届いた際，どの宛先に向けてのリクエストであるかによって

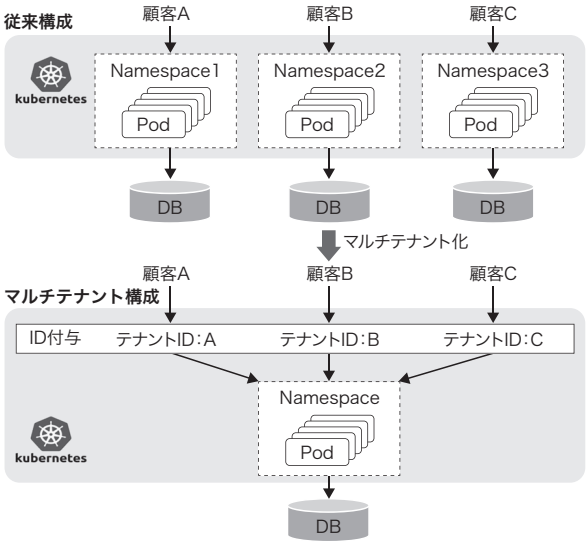

図1. HABANEROTSにおけるマルチテナント環境の構築

マルチテナント化で，運用対象のシステムが一つになり，運用コストを削減できる。

Multitenant environment on HABANEROTS

顧客を識別するためのID（識別子）を割り当て，それを一つのNamespace内のPod群で処理することにした。この変更により，特に市場初期やサービスの初期段階で，顧客ごとにインスタンスのリソースを無駄にせず，効率的にインスタンスを活用できるようになる。また，運用対象のシステムが一つになることにより，運用自体のコストも軽減される。

2.1 マルチテナント設計

前述したように，マルチテナント構成に変更する場合は，複数のテナントが一つのシステムを共有するため，テナント間での独立性が非常に重要となる。例えば，テナント間で，あるビジネスに関する情報がほかのビジネスに関する情報の中に紛れ込んだりすると非常に重大なインシデントとなるし，ほかのテナントにおける処理の影響を受けて性能面で大きく劣化すると，使いづらいものになってしまう。このため，(A)機能面でのテナント間分離，(B)性能面でのテナント間分離，の二つの面でその要件を確認する。

今回は，HABANEROTSのマイクロサービスアーキテクチャーを活用してマルチテナント設計を行った。マイクロサービスアーキテクチャーは，システムを小さな独立したサービスに分割し，それぞれが特定のビジネス機能やユーザー要件を担当する。各サービスは，独自のDB（データベース）を持ち，API（Application Programming Interface）を介してほかのサービスと通信する。このため，各マイクロサービスの独立性と自治性が特長であり，分散性とスケーラビリティーも提供する。

HABANEROTSは，リソースごとにサービスを分割してシステムを構成しており，マルチテナント化では，特にこの独立性を活用して，リソースごとに顧客の求める性能とコストのバランスが取れるように設計する。具体的には，(A)の機能面は必ず完全に分離されている必要があるが，(B)の性能面は，そもそもマルチテナント構成でなくても完全に分離されておらず，具体的な利用シーンによって，どのぐらいのコストを掛けて，どの程度の分離を保証するのが最適であるかが変わってくる。

そこで，これまでの運用実績から，時系列データリソースとファイルリソースについて，特に調整を行った（**図2**）。

2.2 時系列データリソース

時系列データリソースは，バックエンドとして使用しているDBに負荷が集中するため，テナント間で共有せずに別のインスタンスのままとする案も有力である。しかしその場合，性能面での要求は容易に満たせるものの，当初の目的であるテナント当たりのコスト削減に対しては大きなマイナス要素となる。このDB部分は，既存顧客の運用傾向から，利用時に大きな負荷が掛かり得る。そこで，利用機会が少ない集合演算機能について，当初はマルチテナントで提供せず，以後の要望によって機能解放を検討することで，コストを抑えつつ性能の維持を達成した。

2.3 ファイルリソース

ファイルリソースは，巨大なファイルの転送実行時に課題があった。これまでは，リクエストごとに標準的なライブラリーを用いて，ある程度大きなバッファーと一時的なストレージを利用していた。このため，マルチテナント化した場合，特に巨大なファイルの場合にはメモリーとストレージの使用量が急増し，システムのパフォーマンスに影響を与える問題があった。

これに対処するため，内部アーキテクチャーを見直し，ストレージの一時的な利用を減らすことでメモリー使用量を抑えるとともに，リクエストのバッファーサイズを適切に制御することでメモリー使用量を最適化した。これにより，リクエストの滞留時間と滞留中のメモリー使用量を低減し，システム全体のパフォーマンスを向上させることができた。

このアプローチにより，巨大なファイルの転送がほかの処理に影響を与えることなく，性能の現実的な分離が実現できた。

2.4 その他のリソース

その他のリソースは，特に性能上の懸念点が見られないことから，一つのテーブルにテナントID保存用カラムを追加する，一般的な方法でデータの分離を行った。この手法は，2.3節のファイルリソースでも，マイクロサービスの改善後に適用した。

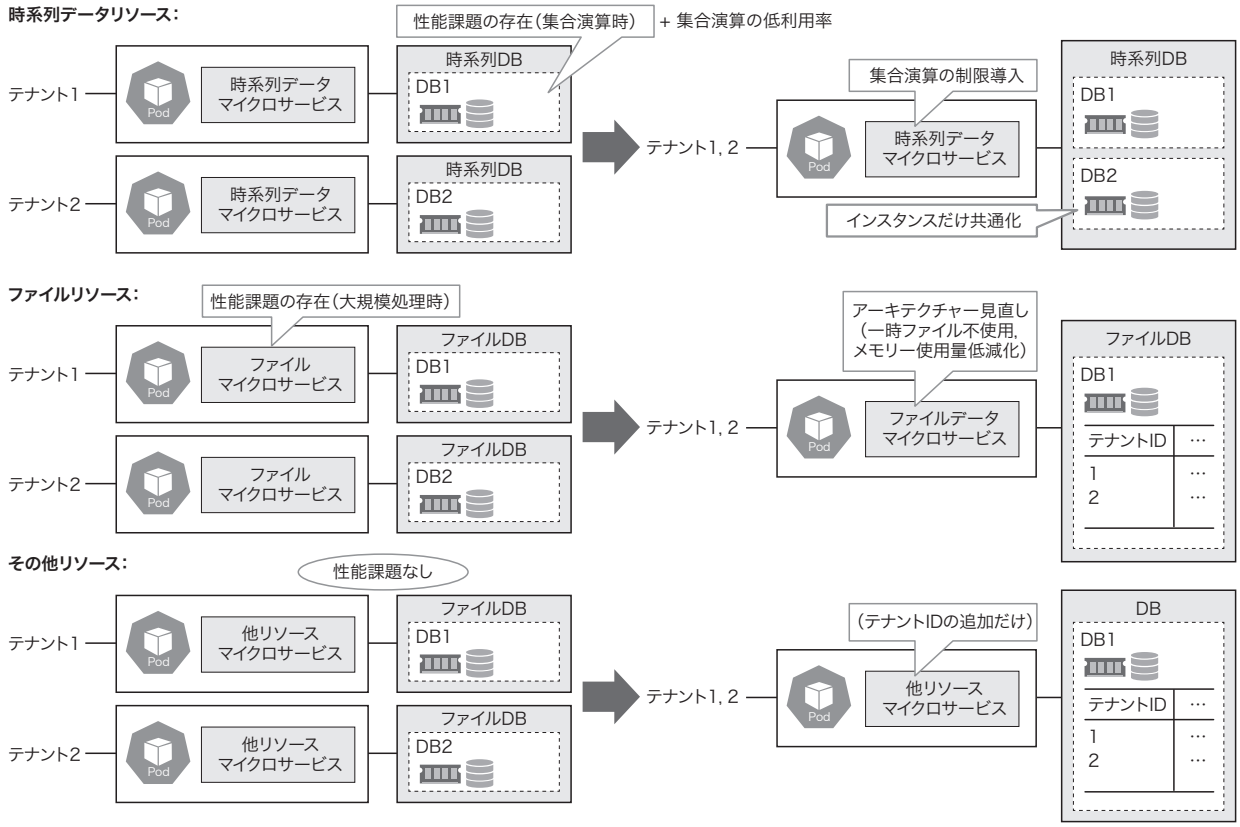

図2. 各リソースの対応方法

マイクロサービスの独立性を活用し，リソースごとに求める性能とコストのバランスが取れるように設計している。

Resource management policy to secure independence for each tenant

このように，マイクロサービスアーキテクチャーの利点を生かし，各リソースの問題と顧客の使用方法を基に，それぞれに適したマルチテナント化を実施した。

また，これまでは，個別にインスタンスを作成して割り当てていたため問題が表面化していなかったが，マルチテナント化では，契約以上のリクエスト要求を処理してしまうとほかの顧客に影響を与えるおそれがある。このため，仕様で定義されているリクエスト数を大幅に超えるアクセスが来た場合には，サーキットブレーカーを発動するような更新も行った。

3. ホスティング環境のセルフサービス化

顧客の利便性向上や，HABANEROTS運用担当者の運用コスト削減のため，ホスティングアプリケーションの顧客に対し，割り当てられたNamespace内でのデプロイメントやバーチャルサービスの編集権限を許可し，自身での各種操作が可能なセルフサービス化を行った。デプロイメントの編集権限により，顧客自身でアプリケーションのコンテナイメージ更新や環境変数の設定が可能となり，バーチャルサービスの編集権限により，アプリケーションへのルーティング条件の設定やHTTP（Hypertext Transfer Protocol）応答ヘッダーの追加が柔軟に行えるようになった。しかし，顧客の操作ミスや悪意のある操作により，ほかの顧客のアプリケーション動作に影響を及ぼしたり，契約で定めた利用範囲を超えたアプリケーション動作をしたりといった新たなリスクが生じるおそれがあるため，次のような対策を行った。

デプロイメントの編集権限では，ホストマシン上のファイルへのマウントが可能になることや，ホストマシン上のルートユーザーと同様の権限を持つ特権コンテナでPodの実行を行えるセキュリティーリスクが生じる。

これを防ぐために，Kubernetesの機能であるPSA（Pod Security Admission）を用いて制限を行った。PSAは，Podのセキュリティー設定がセキュリティーポリシーに従っているかどうかをチェックし，従っていないPodの実行を禁止するものである。セキュリティーポリシーは，あらかじめ用意された，三つの異なるセキュリティーレベルのポリシーから選択できる。HABANEROTSでは，ホスティング環境のNamespaceに三つのうち中間のセキュリティーレベルである

ベースラインを適用し，セキュリティーリスクを伴うPodの実行を禁止した。

そのほかにも，デプロイメントの編集権限により，Podに対するCPUやメモリー使用量の上限を変更できる。顧客が自由にCPUやメモリー使用量の上限を大きくすると，コンテナをホストするマシンのCPUやメモリーの逼迫（ひっぱく）やコストの増加につながる。そこで，ResourceQuotasと呼ばれる機能を用いて，顧客のNamespace単位で使用可能なCPUやメモリー使用量に制限を設けるようにした。

バーチャルサービスについても，次のような理由でほかの顧客のアプリケーションに影響を及ぼすリスクがある。バーチャルサービスは，指定したゲートウェイからの外部トラフィックをサービスメッシュ内の特定サービスにルーティングするルールを記述できる。ゲートウェイは，外部トラフィックのエントリーポイントである。通常顧客のNamespace内のバーチャルサービスでは，そのNamespaceのゲートウェイを指定すべきだが，編集権限により，別のNamespaceのゲートウェイを指定できてしまう。このような設定は，別のNamespaceのゲートウェイを通るリクエストのルーティングに影響を与え，ほかの顧客のアプリケーションに影響を及ぼすリスクがある。

この問題を解決するために，OPA（Open Policy Agent™）[5]を導入した。OPAは，オープンソースのポリシーエンジンで，Kubernetesで実行可能なリソースに関する制約を，Rego言語で記述できる。Kubernetesではマニフェストと呼ばれるyamlフォーマットのファイルでリソースを定義するが，Rego言語は，このyamlの構造に対して制約を自由に記述できる。前述した編集リスクを禁止するため，バーチャルサービスを適用するゲートウェイは顧客のNamespaceで定義されたもの以外を許容しないような制約をRego言語で記述した。

4. あとがき

当社は，HABANEROTSのマルチテナント化を導入し，リソースごとにビジネス間のデータを分離することで，管理コストの削減とサービス設定の迅速化を図った。更に，ポリシーベースの管理を導入することで，セキュリティーを担保しながらホスティング環境のセルフサービス化を行い，管理コストの削減と柔軟なサービス設定を実現した。

これらの取り組みにより，低コストで効率的なCPSサービスの提供が可能となり，当社の競争力強化に寄与することが期待される。今後も，継続的なシステムの最適化と改善を行うことで，より高度なサービスの提供を目指していく。

文　献

(1) ジリエ アルマン．HABANEROTSを基盤としたTOSHIBA SPINEX Marketplaceの構築．東芝レビュー．2023，**78**，4，p.38–41．<https://www.global.toshiba/content/dam/toshiba/jp/technology/corporate/review/2023/04/f04.pdf>，（参照 2024-02-29）．
(2) 東芝エネルギーシステムズ．“SaaS版の「TOSHIBA SPINEX for Energy」提供開始”．ニュースリリース．<https://www.global.toshiba/jp/news/energy/2024/02/news-20240201-01.html>，（参照 2024-02-29）．
(3) 内田正之，樋口靖和．CPSサービスの迅速な立ち上げに貢献する東芝IoT基盤サービス HABANEROTSにおけるサービスメッシュの活用．東芝レビュー．2020，**75**，5，p.31–34．<https://www.global.toshiba/content/dam/toshiba/migration/corp/techReviewAssets/tech/review/2020/05/75_05pdf/a09.pdf>，（参照 2024-02-29）．
(4) The Linux Foundation. "Production-Grade Container Orchestration". Kubernetes. <https://kubernetes.io>,（accessed 2024-02-29）.
(5) The Linux Foundation. "Policy-based control for cloud native environments". Open Policy Agent. <https://www.openpolicyagent.org>,（accessed 2024-02-29）.

樽家　昌也　TARUI Masaya
デジタルイノベーションテクノロジーセンター 技術開発室
サービスプラットフォーム開発部
情報処理学会会員
Service Platform Development Dept.

望月　翔平　MOCHIZUKI Shohei
デジタルイノベーションテクノロジーセンター 技術開発室
サービスプラットフォーム開発部
Service Platform Development Dept.