# モダンアプリケーションアーキテクチャーの採用で顧客の要求に柔軟に対応できるポイント管理システムPointArtist2

**PointArtist2 Point Management System with Flexibility to Meet Requirements of Individual Users Applying Modern Application Architecture**

長谷川　秀司 HASEGAWA Shuji　　藤本　正人 FUJIMOTO Masato

小売業界では，消費者の購買意欲の促進を図るなどの目的で，ポイントサービスを導入するケースが増えている。

東芝デジタルソリューションズ（株）は，これまで小売業向けのポイント管理システムPointArtistを提供してきた。近年，電子商取引（EC）やキャッシュレス決済などの普及で小売業を取り巻く環境は大きく変化し，これに合わせて必要な機能を柔軟に提供できるマネージドサービスの実現が望まれていた。そこで，従来型のモノリシックなアーキテクチャーに代えて，モダンアプリケーションアーキテクチャー（注1）を採用することで，マイクロサービスとコンテナにより，柔軟かつ短期間に機能変更ができるPointArtist2を新たに開発し，提供を開始した。

The movement toward the introduction of point services has recently accelerated in the retail industry aimed at increasing consumer demand though point collection programs.

Toshiba Digital Solutions Corporation has been providing the PointArtist point management system to retail companies. However, as a result of the changes that have taken place in the environment surrounding such retail companies with the dissemination of e-commerce and cashless payment systems in recent years, the need has arisen for managed services to flexibly provide functions required for individual point services. We have now developed and launched PointArtist2 as a successor to PointArtist. Through the application of microservices and containers, as well as a modern application architecture instead of the conventional monolithic architecture, PointArtist2 makes it possible to swiftly and flexibly change service functions.

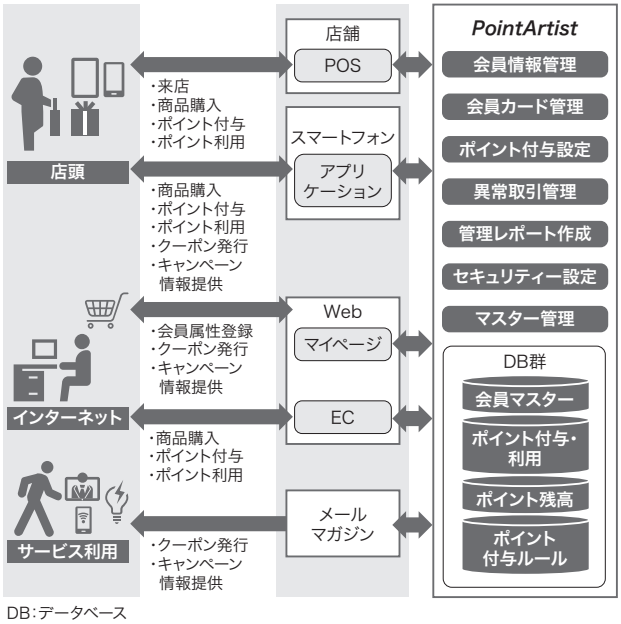
## 1. まえがき

スーパーマーケットや，ショッピングセンター，百貨店などを経営する小売業界では，消費者の来店頻度の増加や購買意欲の促進を図る施策の一つとして，ポイントサービスを導入することが多い。東芝デジタルソリューションズ（株）は，小売業界向けに，POS（販売時点情報管理）システムと連携し，購入時にポイントを付与するポイント管理システムPointArtist（**図1**）を提供してきた。

近年，インターネットやモバイルネットワークの普及と拡大に伴い，ECの増加や，スマートフォンを利用したプロモーションに代表される販売促進手法の多様化，共通ポイント・キャッシュレス決済ポイントの急速な普及など，小売業界の市場環境が加速度的に変化している。

また，当初，小売業界から始まったポイントサービスは，公共サービスや行政サービスなどにも広がり，業界・官民を問わず幅広い領域で活用されるようになってきた。一方，

（注1） システムを，マイクロサービスといった小規模なサービス志向プロセスの組み合わせで構成し，システムの構築や運用の効率化を図る手法。




DB：データベース

**図1. PointArtistの概要**

小売業向けに，POSなどと連携して，ポイントをリアルタイムに管理するシステムである。

Overview of PointArtist point management system

独自のポイントサービスによる優良顧客の維持や，ポイント付与に掛かるコストの削減など，ハウスポイント（企業が自社で付与するポイント）の需要も大きい。

そのため，新たなポイント機能の追加や，様々なPOS端末・機器への対応，ECとのポイントの統合，スマートフォンを使用した各種サービスとの連携など，多様な要求が発生するようになり，それらに応えるため，必要なときに機能変更を柔軟かつ短期間で行えるシステム環境が求められている。

今回，顧客の要求に合わせて必要な機能を提供するマネージドサービスを実現するため，モダンアプリケーションアーキテクチャーを採用して柔軟な機能変更を可能にしたポイント管理システムPointArtist2を新たに開発し，提供を開始した。ここでは，PointArtist2の機能と特長について述べる。

## 2. PointArtistの課題

従来の多くのシステムと同様に，PointArtistは，複数の機能を組み合わせて一つのシステムとするモノリシックな構造になっていた。

そのため，機能変更範囲の大小にかかわらず，システム全体への影響を配慮した調査・設計，及び全機能のテストを行う必要があった。長年，機能変更を繰り返した結果，アプリケーション資産の肥大化とともにシステムの複雑さが増し，影響調査範囲・テスト範囲が拡大して，膨大な手間と時間が掛かる状況となっていた。

また，変更後のアプリケーションのリリースも，各機能が密接に結合したモノリシックな構造のために，システム全体を停止した上で実施する必要があり，運用性の改善が求められていた。

加えて，耐障害性や性能は動作環境に依存する部分が多いため，それらの要求を満たすプラットフォームを設計・構築する必要があった。

## 3. モダンアプリケーションアーキテクチャー

PointArtistの限界を打破し，顧客の要求に合わせて機能を提供するため，モダンアプリケーションアーキテクチャーを採用して，PointArtist2を新たに開発した。PointArtist2の主な動作環境であるAWS（Amazon Web Services）を例に，モダンアプリケーションアーキテクチャーの実装について述べる。

### 3.1 コンテナ

コンテナとは，アプリケーションを標準化したモジュールで構築し，どのようなコンピューティング環境にも容易に導入できるようにする技術である。PointArtist2はアプリケーションをコンテナで動作させることにより，水平スケール調整と環境構築の容易さ，及びリソースの有効活用を実現している。Docker（コンテナの仮想環境を提供するプラットフォーム）が動く環境であれば，個人で使用しているパソコンであってもPointArtist2の動作が可能である。

コンテナを運用に耐えられる実装とするには，コンテナを管理するオーケストレーションが必要である。AWSのオーケストレーションには，ECS（Elastic Container Service）とEKS（Elastic Kubernetes Service）の二つのサービスがある。当初は，最近注目されているオーケストレーションシステムであるKubernetesを使用するEKSの採用を考えていた。しかし，Kubernetesは3か月ごとにバージョンアップされており，EKSもこの頻度で更新される。バージョンが頻繁に更新されること，またリリースされたバージョンのサポート期間が約12か月であることはPointArtist2のメンテナンス性，運用性を阻害することから，ECSを採用することにした。

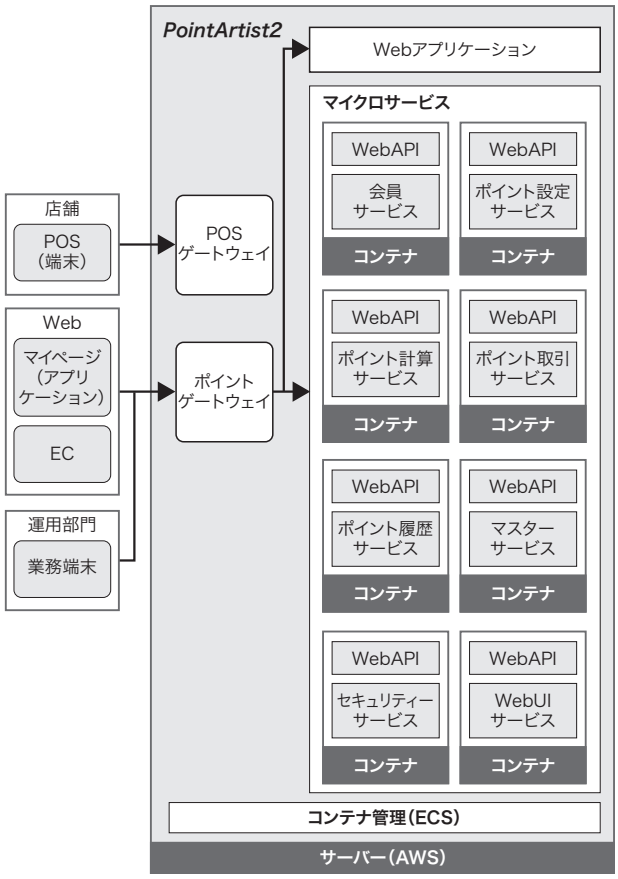
コンテナを動作させる環境としては，EC2（Elastic Compute Cloud）とFargate（コンテナ向けサーバーレスコンピューティング環境）がある。これらの中から，OS（基本ソフトウェア）やミドルウェアのバージョンアップ作業が不要で，オートスケーリングが容易なFargateを選択した。

### 3.2 マイクロサービス

PointArtist2は，機能間の疎結合化，及び機能変更時の影響範囲の極小化を目的に，マイクロサービスアーキテクチャーを採用している。これは，システムの機能を，マイクロサービスと呼ばれる比較的小さなシステムに分割して構築し，それらのマイクロサービスを組み合わせることで，必要な機能のシステムを構成するアーキテクチャーである。マイクロサービスの設計では，その粒度の決め方が重要である。

PointArtist2では，各機能が更新するテーブル（データベース（DB）上の表）を洗い出して同じテーブルを更新する機能をまとめ，ここでまとめられた機能群をマイクロサービスの最小粒度とした。これにより，更新時にトランザクション（一連のデータ処理の単位）の一貫性を保ち，マイクロサービスの独立性を高めながら整合性の取れたデータ更新が可能になった。また，PointArtistでは，負荷の掛かる検索処理とPOSからの更新処理が同じDBを使用していたため，十分な更新処理速度で実行できないことがあった。そこで，PointArtist2では，大量データの検索処理用に読み取り専用レプリカを用意し，更新処理を高速化できた。

このような過程を経て，最終的にPointArtist2を8個のマイクロサービス（**図2**）で構成した。今後，機能を変更するときは，独立したサービスの追加・分割・統合を実施して




WebUI：Webユーザーインターフェース

**図2.　PointArtist2のマイクロサービス構成**

モダンアプリケーションアーキテクチャーを採用して，8個のマイクロサービスで構成している。必要な機能を柔軟に組み合わせて，マネージドサービスとして提供できる。

Overview of PointArtist2 point management system consisting of microservices and containers

いく。各サービスのインターフェースの不変性を維持することで，インターフェースの呼び出し側が影響を受けない作りとした。

### 3.3　機能のAPI化

マイクロサービス化に伴い，バッチ処理を除く業務ロジックは全てWebAPI（Web Application Programming Interface）で実装した。各機能はWebAPIを呼び出すことで動作し，画面表示アプリケーションは入力や値のチェック，結果の表示などを行い，WebAPIを呼び出すことで処理している。WebAPIはREST（Representational State Transfer）モデルとし，当社のフレームワークであるStaveware Core V6（フレームワークであるSpringをベースとする）を使用した。画面表示アプリケーションはSPA（Single Page Application）で実装し，フレームワークにはVue.js（ユーザーインターフェース（UI）を構築するためのJavaScriptフレームワーク）を使用している。

機能変更や新機能作成は，WebAPIの拡張，若しくはWebAPIを組み合わせたWebAPIの作成により実装する。このため，販売促進のための新たなポイント機能の追加や，分析システムなどほかのシステムとのデータ連係，スマートフォンなどを利用する様々なサービスとの連携など，幅広い対応が可能である。極端な例では，WebAPIを公開するだけで，顧客側で自由に機能変更や画面表示アプリケーションの開発が可能である。

また，PointArtist2は，POSから取り引きデータを受信してポイント付与する使い方が多い。このとき，POSとのインターフェースが重要である。POSとのインターフェースはPOSのベンダーによって様々であり，通信方式やデータフォーマットも異なる。POSからPointArtist2で準備しているWebAPIを呼び出すように改修することも可能であるが，一般にPOSの改修コストは高い。PointArtist2では，POSゲートウェイという方式でこの問題を解決する。POSゲートウェイは，POSとの通信方式やデータフォーマットの変換を行い，PointArtist2が準備しているWebAPIを呼び出す機能である。POSベンダーごとにPOSゲートウェイを作成することで，業務ロジックを変えずに様々なベンダーが保有するPOSとのインターフェースが可能になる。

### 3.4　バッチ処理とログ出力処理

バッチ処理はJavaで作成し，その起動に，Step Functions（分散アプリケーションやマイクロサービスのコンポーネントを容易にコーディネートできるAWSのマネージドサービス）を使用する。処理時間が短ければ，Lambda（サーバーレスコンピューティングを行うAWSのマネージドサービス）の利用も考えられるが，大量のデータを扱う処理が多いため，Step FunctionsでJavaプログラムを起動することで，処理時間の影響をなくした。

ログ出力は，CloudWatch（リソースやアプリケーションログを監視できるモニタリングサービス）に出力する。ただし，直接出力するのではなく，Fluentd（オープンソースのデータコレクター）を経由して，CloudWatchに書き込むようにしている。これにより，Fluentd設定変更で，顧客の環境に合わせて，CloudWatch以外のDataDog（Datadog社のSaaSベースの監視アプリケーションサービス）やElasticsearch（Elastic社の分散処理マルチテナント対応検索エンジン）などに，ログを転送することも可能となる。

## 4.　サービス運用面の改善

PointArtist2は，コンテナやマイクロサービスなどのモダンアプリケーションアーキテクチャーを採用することで，サー

ビスの運用面での改善を実現した。

### 4.1 DevOps，CI/CD

PointArtist2では，Jenkins（オープンソースソフトウェアのCI（継続的インテグレーション）ツール）を使用して，ビルド及びリリース作業を自動化している。プログラムソースを社内のリポジトリーに置いているため，サーバーでオブジェクトのビルド及びコンテナイメージを作成した後，ECR（Elastic Container Registry）に送信する。併せて，ECSを使用しているため，プログラムを更新したコンテナを起動し，更新前のコンテナを停止することにより，順次プログラムが更新される。このため，システムを稼働させたままプログラムの入れ替えができ，リリース作業を含めてシステム停止の必要がない。

このように，ソフトウェアの開発と運用を連携させたDevOps，及びCI/CD（継続的デリバリー）が実現できる。

### 4.2 自動復旧

PointArtist2では，マイクロサービスにより各機能間の疎結合を実現しており，一つのマイクロサービスがほかのマイクロサービスの機能に影響を与えることはない。しかし，システム全体としては各マイクロサービスが連携していることが前提であり，一つのマイクロサービスの障害がシステム全体の機能不全をもたらすことになる。このため，システムの機能を維持するには，障害が発生したマイクロサービスの速やかな復旧が欠かせない。

PointArtist2は，各マイクロサービスを監視しており，障害を検知したときは自動復旧させることで，システム全体の機能不全を最小限にとどめている。

### 4.3 オートスケール

PointArtist2だけでなく各種システムは，一般に，必要とされる性能とリソースのバランスを取る必要がある。従来は，必要とされる性能のピークに合わせて，十分なリソースを用意していた。

PointArtist2では，各コンテナのリソース使用状況を監視し，リソース使用がしきい値を超えた場合に，新たなコンテナを起動させて負荷が掛かっているマイクロサービスの処理を分散させることで，性能拡張を実現している。また，処理が減少した場合は，コンテナを停止することも可能である。これにより，必要とされる性能とリソースの自動調整を行うオートスケールに対応した。

## 5. あとがき

PointArtist2は，モダンアプリケーションアーキテクチャーを採用することで，使いたいマイクロサービスの選択的利用や，WebAPIの組み合わせによる必要機能の実現，システム停止不要のプログラムリリースなど，必要機能の柔軟な取捨選択を実現した。また，サービス運用面の改善により，自動復旧による障害への対応，及びオートスケールによる性能要求への対応が可能になった。これらにより，顧客の要求に合わせて機能を提供するマネージドサービスを実現した。また，マネージドサービスにすることで，顧客はサーバーなどの維持・管理が不要になるというメリットもある。

現在は，既存のシステムとの親和性を確保できるサブスクリプションライセンス（特定期間内のライセンス契約）として提供している。今後，マネージドサービスとしての提供を通して，顧客のデジタルトランスフォーメーションに伴う変化をサポートしていく。

当社は，今後もモダンアプリケーションアーキテクチャー及びマネージドサービスの最新技術を取り入れ，PointArtist2を幅広い領域で活用されるポイント管理基盤として提供していく。


長谷川　秀司　HASEGAWA Shuji
東芝デジタルソリューションズ（株）
ICT ソリューション事業部 流通ソリューション技術部
Toshiba Digital Solutions Corp.

藤本　正人　FUJIMOTO Masato
東芝デジタルソリューションズ（株）
ICT ソリューション事業部 流通ソリューション部
Toshiba Digital Solutions Corp.