アジャイル開発を支えるPaaS基盤

Platform as a Service to Support Agile Software Development

池谷　直紀　　木村　隼人　　藤本　　宏
■IKETANI Naoki　■KIMURA Hayato　■FUJIMOTO Hiroshi

東芝は，不確実性の高いアプリケーション開発を効率的に進めるために，共同作業スペースeXtreme Design Studio（XDS）を開設してアジャイル開発を実践するための開発基盤を用意した。この開発基盤は，Cloud Foundry(†)を用いて構築されたPlatform as a Service（PaaS）基盤であり，アプリケーション実行環境を提供し，ミドルウェアを要求に応じて利用できるようにする。また開発成果物を容易に配備できる仕組みにより，長期安定動作用や，受入れ検査用，デモンストレーション用など各種バージョンの複数のアプリケーションを一つのPaaS上で動作させることもできる。このPaaS基盤の活用によりアプリケーション開発者（以下，開発者と略記）がコード開発に専念できるようになるとともに，周辺ツールと連携した継続的インテグレーション（CI）環境を構成することによりアジャイルかつ継続的なアプリケーションのリリースが可能になる。このPaaS基盤を活用したIoT（Internet of Things）アプリケーションの試作を行い，開発基盤としての有用性を確認した。

Toshiba has established an application development platform applying agile software development that can improve the efficiency of software development with high uncertainty. This platform is provided as a Platform as a Service (PaaS) based on Cloud Foundry(†), an open source cloud computing PaaS, which allows application developers to provide middleware on demand as well as application execution environments. Application developers can easily use the necessary middleware via the network and simultaneously implement several versions of an application on a single PaaS platform, such as an official release version, an acceptance test version, a demonstration version, and so on. This reduces the burden on the developers of constructing and managing the development environment, allowing them to concentrate on the coding work and rapidly release application programs in conjunction with utility tools for continuous integration (CI). We have confirmed the effectiveness of this PaaS for agile software development through the results of trial production of an Internet of Things (IoT) application.

1　まえがき

現在，アプリケーションの開発ライフサイクルは非常に短くなってきている。ビジネスの変化に追従するため，月曜日に考えたアイデアを金曜日にサービスとしてリリースするようなスピード感が求められる。またリリースしたサービスも，常に改良と機能追加をし続けることも求められる。

開発効率を向上させるための施策として，開発者が本来作業すべきロジック開発行為（コーディング）に集中できるように，開発環境や試験環境の準備などの付帯作業から解放し，かつこれらの作業を短縮していくことが必要になる。

今回，開設した共同作業スペースeXtreme Design Studio（XDS）での開発サービス提供にあたって，開発者が開発行為に専念できるようにするため，開発環境や試験環境の準備を自動化する開発基盤を用意した。これはPlatform as a Service（PaaS）と呼ばれる機能を提供する。

2　PaaSの機能とその効果

2.1　PaaS登場の背景

従来のアプリケーション開発では開発ごとに開発環境や試験環境を調達し構築していた。アプリケーションの仕様に合わせてサーバや，ストレージ，ネットワーク機器などのハードウェアを用意し，構成設計を行ったうえで必要なミドルウェア群をセットアップして，初めてアプリケーションの開発と評価に取り掛かれることになる。これらの準備には数か月間掛かることもあり，開発者が自分のアイデアを実現するまでに多くの時間を要するうえに，本来取り組むべき開発以外の作業に時間を取られてしまうことになる。

そこで，これらの機材やミドルウェアを個別に用意せず，あらかじめ構築され設定された環境を，必要なときにネットワークを介して必要なだけ使うという新たな利用形態が現れてきた。

2.2　PaaSの機能と利用形態

PaaS基盤は，登録されたWebアプリケーションを実行する基盤であり，ミドルウェアやリソース（メモリ，ストレージ）などのプラットフォームをあらかじめ決められた構成で用意し，アプリケーションから利用できるという形態を提供する。

提供されるプラットフォームはPaaS提供者によってカスタマイズ可能であり，Java(†)やRubyなどの開発言語，Springなどのフレームワーク，Webサーバ，及びデータベース（DB）やメッセージングなどのミドルウェアが，用途や目的に応じた構成で事前に準備されている。

PaaSを利用する開発者の視点では，開発したアプリケーションを動作させるために必要な手順は以下のようになる。

(1) PaaSのアカウント作成を申し込み，PaaS管理者に権限などの設定をしてもらう
(2) 動作するURL（Uniform Resource Locator）を指定してアプリケーションをPaaSに送信する
(3) アプリケーションを動作させるために必要なDBなどのミドルウェアインスタンスを生成し接続する
(4) アプリケーションの動作開始を指定する

この手順において，開発者はその実行環境を構成するハードウェア，仮想マシン，及びOS（オペレーティングシステム）については隠蔽されており，意識する必要がない。Infrastructure as a Service（IaaS）が仮想マシンを払い出すサービスであることと対比すると，PaaSは抽象化されたミドルウェア及びアプリケーション実行環境を利用させるサービスであると言える。

2.3 開発基盤としてのPaaS利用の効果

開発者がPaaSを利用する効果として，以下が挙げられる。

(1) サーバや，ストレージ，OSの準備とPaaSで提供されるDBなどミドルウェアの構築が不要
(2) 同一あるいは異なるアプリケーションを一つのPaaS上で同時に複数実行可能
(3) 実際のサービス提供用のPaaSと共通の構成にすることで，実際のサービス提供時と同等の環境で開発・検査可能

これらにより開発者は自分の作りたいアプリケーションの開発に集中でき，よりスピーディな開発を実現できる。PaaSを利用して開発したアプリケーションは，アプリケーション単体で利用される場合や，開発時とは異なるPaaS上のサービスとして提供される場合がある。その場合には，開発用のPaaSを実用時と共通の構成とすることにより，同等の可用性や運用性を利用でき，更に検査もできるので，開発及び検査時の大きな利点となる。

インフラ管理者の視点でPaaS基盤を活用する効果として，以下が挙げられる。

(1) PaaS基盤にリソースを集約することで組織全体での計算機リソース効率の改善
(2) フレームワーク，ミドルウェア，及びライブラリをPaaSで利用可能な部品として整備・共有可能
(3) OSやミドルウェアの運用及び保守をPaaSの運用及び保守に一元化することで，運用及び保守のコストを削減

3 PaaSを構成する技術

Cloud Foundry(†)はOpen Source Software（OSS）として開発されているPaaS基盤構築ソフトウェアである(1)。主要な機能と特徴を以下に述べる。

(1) アプリケーションの動作及び管理機能
(2) アプリケーションの高可用化と，スケールアウト対応
(3) 単一障害点のない自律分散型アーキテクチャ
(4) Buildpackの追加により実行環境をカスタマイズ可能
(5) ミドルウェアなどのサービスを追加登録可能

3.1 アプリケーションの動作及び管理機能

Cloud Foundry(†)は，登録されたアプリケーションを，指定されたURLでアクセス可能な状態として動作させる。基本的な機能として，アプリケーションの起動と停止，監視と復旧，及びログ収集の機能を提供している。また，複数の異なるアプリケーションを一つのPaaS基盤上で動作させることができる（**図1**）。同一あるいはバージョンの異なるアプリケーションを異なるURLで独立に動作させることもできるので，旧バージョンを動作させながら，開発した新バージョンについて，結合検査用，受入れ検査用，及び安定動作用と段階を分けて動作させることが容易であり，一般的である。

3.2 アプリケーションの高可用化と，スケールアウト対応

Cloud Foundry(†)は，アプリケーションを高可用化させるため，またサーバ数の増大による性能改善すなわちスケールアウトさせるために，以下の仕組みを持つ。

(1) アプリケーションを指定されたインスタンス（実体オブジェクト）数で動作させる
(2) 内部にアプリケーション監視・復旧機能を持ち，指定のインスタンス数が正常に動作していない場合には，アプリケーションインスタンス追加などの適正化を行う

この仕組みにより，アプリケーションを例えば三つのノードで動作させてロードバランサからのアクセスを割り振るように構成することで，負荷を分散させるとともに，単一サーバ障害に対して耐性のある構成が容易に実現できる。また高負荷時やメモリなどのリソース不足になりそうな際に，インスタンス数

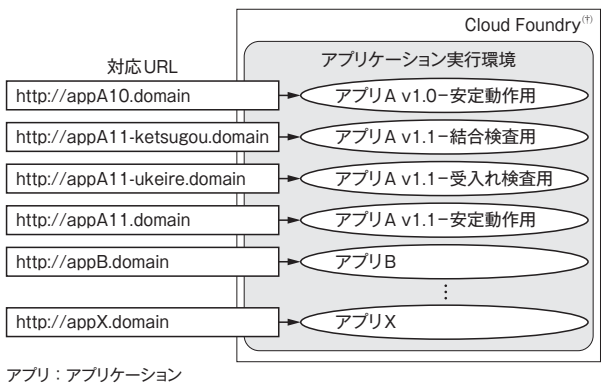

図1. URLとアプリケーションの関係 ― Cloud Foundry(†)上では同一あるいはバージョンの異なるアプリケーションを動作させることができ，それぞれを異なるURLで動作させられる。

Relationship between uniform resource locators (URLs) and applications on cloud computing PaaS

一般論文

を増やして回避することも設定だけで実現可能である。

3.3　単一障害点のない自律分散型アーキテクチャ

Cloud Foundry(†)は単一障害点のない疎結合のモジュール群による自律分散型アーキテクチャとなっている（**図2**）。

基本的な仕組みは以下のとおりである。

(1)　アプリケーションはDEA（Droplet Execution Agent）と呼ばれる実行環境で動作

(2)　アプリケーションに対するHTTP（Hypertext Transfer Protocol）でのアクセス要求はルータが適切にDEA上のアプリケーションに割り振る

(3)　ヘルスマネージャがアプリケーション状態を監視，制御

(4)　クラウドコントローラがモジュール群の動作を制御

(5)　NATSと呼ばれるメッセージバスを介してモジュール群が連携

(6)　外部サービスとの接続はルータを介して行われる

ここに登場する各モジュールは冗長化して動作させることができ，またその状態監視と復旧を行う仕組みも内包している。この仕組みによりPaaS基盤は，例えばサーバ故障などの単一障害に対する耐障害性を持つ基盤として動作する。

3.4　Buildpackの追加による実行環境のカスタマイズ

Cloud Foundry(†)は，Buildpackという実行環境定義の追加によりJava(†)，Ruby，Pythonなどの多様な言語やフレームワーク向けに作られたアプリケーションを動作させられる。PaaS基盤運用者は基盤としてサポートする実行環境を，Buildpackを利用して，必要に応じて追加することができる。

3.5　サービスの追加登録

Cloud Foundry(†)は，サービスをカタログ化してアプリケーションから接続できるようにする。例えばDBサービスや，翻訳サービス，携帯電話通知サービスなどのミドルウェアやツールをサービスとして登録し，アプリケーションから利用することができる。

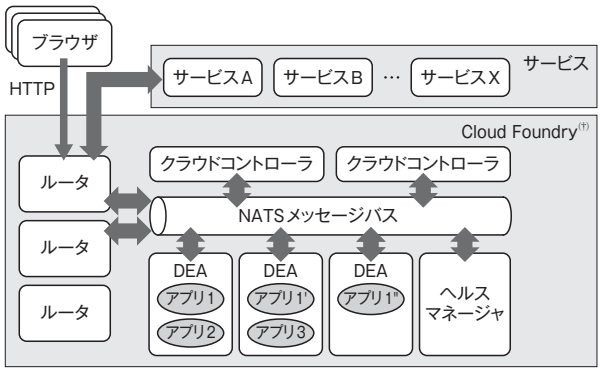

図2．Cloud Foundry(†)のアーキテクチャ ― Cloud Foundry(†)は疎結合の複数のモジュールで構成されており，単一障害点がない基盤を構築できる。アプリケーションは内部のDEAと呼ばれる実行環境で動作する。

Architecture of Cloud Foundry(†)

開発者がサービスを利用する手順は以下のようになる。

(1)　カタログから選択したサービスのインスタンスを生成
例えばDBなどのサービスでは，この時点でアプリケーション用に分離された論理的なサービスが作成される

(2)　アプリケーションに対するバインド（割当て）を行う
サービスに接続するためのURLや必要に応じてパスワードなどが提供される

(3)　アプリケーションを動作させる

手順(2)で提供される，サービスに接続するための情報は，アプリケーション動作時の環境変数として提供される。開発者は環境変数を用いてサービスを抽象化して開発しておくことで，異なるサービスインスタンスへのバインドがアプリケーションの変更なしで可能となり，アプリケーションを異なるCloud Foundry(†)環境で動作させることも容易にできる。

4　PaaSを用いたIoTサービスの開発事例

PaaSを用いたアプリケーション開発事例に基づいてPaaSの導入効果について述べる。

4.1　IoTサービスの概要

ここで述べる事例では，受注前の新規案件のIoT（Internet of Things）サービスを題材として試行的に開発した，数千か所の拠点で管理している機器を遠隔で一元管理するアプリケーション（以下，遠隔管理アプリケーションと呼ぶ）をCloud Foundry(†)上に構築した。各拠点で管理している機器の多種多様なセンサデータがサーバに異なる頻度で送られ，様々な観点から加工された機器情報を可視化する画面や，異常値を検知してアラートを発生する機能などを備えたアプリケーションを開発した。

4.2　Cloud Foundry(†)を用いた開発

この事例は，ステークホルダー間でアプリケーションの仕様について明確にしながらアプリケーションを構築する必要があったため，仕様変更に柔軟に対応できるアジャイル開発(注1)を採用した。開発対象の遠隔管理アプリケーションに接続されるセンサ数やアプリケーションの利用者数の上限が定まっておらず，どの程度リソースが必要になるか見込めなかったため，リソースを柔軟に調整できるCloud Foundry(†)を利用した。アジャイル開発を進めていくうえで，アプリケーションのビルドや，テスト，配備といった一連の作業が継続的に行われるため，**図3**に示すような継続的インテグレーション（CI）環境を構築し，一連の作業を自動化した。

4.3　Cloud Foundry(†)の有効性

Cloud Foundry(†)を用いて開発した結果，以下の有効性を確認した。

（注1）　ソフトウェア機能を利用者に新たな価値を与える最小単位に分割し，短い反復期間を繰り返しながら迅速かつ適応的に開発する手法。

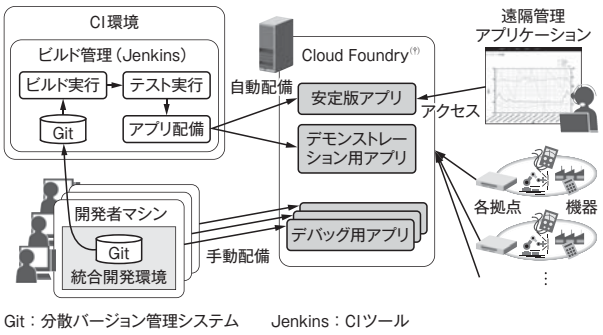

図3. Cloud Foundry(†)を用いたIoTサービスの開発 ― 大量のセンサデータを扱うIoTサービスの開発に，Cloud Foundry(†)を用いたCI環境を利用した。

Development of IoT services using Cloud Foundry(†)

4.3.1　開発時の初期コスト削減　Cloud Foundry(†)にはあらかじめ開発言語のBuildpackやDBなどのミドルウェアが備わっている。遠隔管理アプリケーションは，ミドルウェアとしてApache(†) Tomcat(†)，PostgreSQL(†)，及びSpringベースのJava(†)環境で動作するが，これらの事前環境構築なしで即座に動作確認することができた。そして，Buildpackとして定義された実行環境を利用しているため，ミドルウェアのバージョンによる互換性の問題は発生しなかった。

4.3.2　アプリケーションの可用性を確保するための開発コスト削減　アプリケーションの監視機能やスケールアウトの仕組みがCloud Foundry(†)で提供されている。開発者はPaaS基盤で提供されるこれらの機能を利用することにより，ロードバランサの設定や，アプリケーションのプロセス監視などによるフォールトトレラント機能の実現に掛かるコストを大幅に削減することができた。

4.3.3　開発状況や利用用途に応じて，アプリケーションを柔軟かつ迅速にリリース・更新可能　Cloud Foundry(†)ではアプリケーションの実行環境の構築が簡略化されている。これにより，例えば図3に示したように，開発者はアプリケーションの配備先を切り替えるだけで，最新の安定版や，デモンストレーション用，デバッグ用といった用途に応じたアプリケーションを低コストで用意することができた。

更に，CI環境を利用することでアプリケーションの配備先を切り替えて配備するまでの一連の作業が自動化できるため，アプリケーションのリリースや更新を柔軟かつ迅速に行うことができた。

4.4　事例のまとめ

この事例では，IoTサービスの開発でCloud Foundry(†)を利用した際の有効性について述べた。PaaS基盤が備えるロードバランサや監視機能を用いることで，アプリケーション側で可用性に関する実装の手間を削減できた。

アジャイル開発では，ステークホルダーに応じて動作するアプリケーションを用意し，仕様に変更があれば柔軟に対応することが必要だった。開発者はアプリケーションの配備先を切り替えるだけで，用途に応じた複数のアプリケーションを柔軟に準備できるようになった。更に，CI環境と連携することで，アプリケーションを開発状況や用途に応じて，迅速にリリースできるようなった。

これらの結果より，Cloud Foundry(†)はアジャイル開発を支える重要な基盤と言える。

5　あとがき

不確実性の高いアプリケーション開発を手早く進めるアジャイル開発では，着手してからプロダクトの初期版が動作するまでのスピード，そして改良版を手早くリリースするスピードが重要である。Cloud Foundry(†)を用いたPaaS基盤を構成することにより，開発者がOSやミドルウェアの構成作業から解放されアプリケーション開発に集中できること，実行環境及びミドルウェアを容易に払い出して利用できること，これらにより開発が迅速に行えることを確認した。

現時点ではPaaS基盤の開発環境としての活用を行っているが，開発環境だけでなく最終的なアプリケーションサービス実行環境までを一貫したPaaS基盤として提供することにより，更に効率的な開発及び継続的な提供を実現していく。

文　献

(1) Cloud Foundry Community. Cloud Foundry. Cloud Foundry Homepage. <http://cloudfoundry.org/>, (accessed 2015-09-14).

池谷　直紀　IKETANI Naoki
東芝ソリューション（株）プラットフォームセンター プラットフォームソフトウェアサービス設計部主査。クラウドサービスの設計・開発に従事。情報処理学会会員。
Toshiba Solutions Corp.

木村　隼人　KIMURA Hayato
東芝ソリューション（株）ソリューションセンター ソリューション生産技術部。クラウドサービス及びアプリケーションフレームワークの開発に従事。情報処理学会会員。
Toshiba Solutions Corp.

藤本　　宏　FUJIMOTO Hiroshi
インダストリアルICTソリューション社 IoTテクノロジーセンター ソフトウェア設計技術開発部参事。クラウドシステムを活用したアプリケーションの設計技術の開発に従事。
IoT Technology Center