特集論文

既存パッケージのSaaS化への取組み

野本泰宏*
服部佐次郎*

Approach of Existing Package on Making to Software as a Service

Yasuhiro Nomoto, Sajiro Hattori

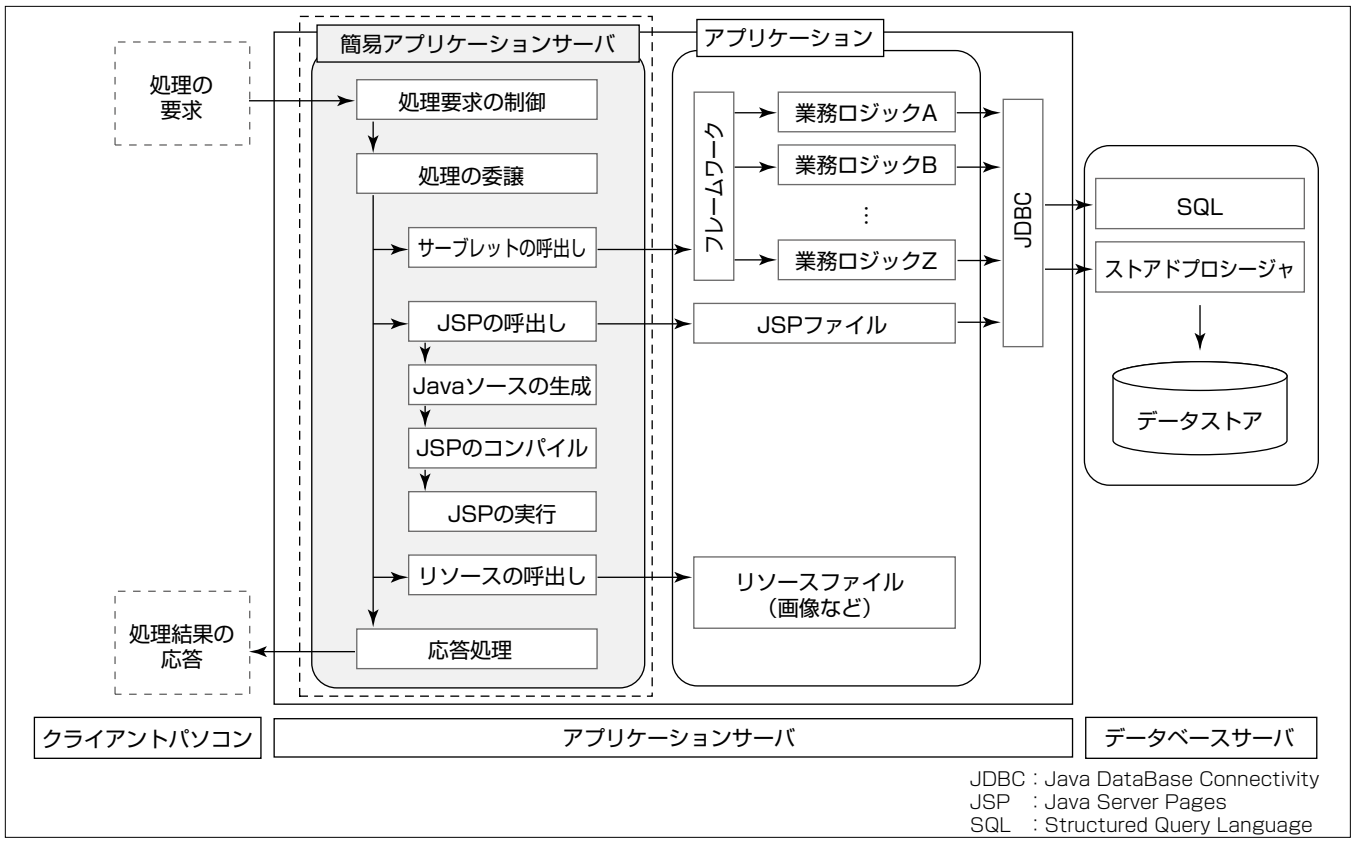
要　旨

近年，ハードウェアや仮想化技術などの進歩を受けて，SaaS(Software as a Service)，PaaS(Platform as a Service)，IaaS(Infrastructure as a Service)のような新しいコンピュータの利用形態が広がり始めており，インターネットを活用したITサービスが急速に拡大してきている。これらの新たな利用形態では，高いスケーラビリティを持つ仮想サーバ群を，“必要なときに”“必要なだけ”サービスとして利用できるところに特長があり，特にSaaSについては，アプリケーションの提供方法に大きな変革をもたらしつつある。

このような動向に対応するため，既存のアプリケーションパッケージをSaaS化するための検討を行い，小売店向け販売管理パッケージによる検証を実施した。このパッケージは，アプリケーションサーバ上で業務ロジックを実行する構造の業務アプリケーションパッケージである。インターネットを介して動作する既存のアプリケーションパッケージの大半はこれと同じ構造を持つが，IaaSなどの実行環境を提供するベンダーによってアプリケーションサーバの仕様が異なっており，SaaS化に向けた移植性が問題となっている。

今回，SaaS化を検討するに当たっては，移植性の問題への対処方法として，アプリケーションサーバ機能の共通化を図ることとし，この実現性を検証するために，IaaSベンダーが提供する仮想化環境上に検証用の“簡易アプリケーションサーバ”を構築して，動作確認を実施した。

その結果，この方式の有効性を確認することができ，同様の構造を持つ業務アプリケーションパッケージを汎用(はんよう)的なIaaS環境下で動作させるための方法を確立できた。



簡易アプリケーションサーバによるアプリケーションソフトの動作イメージ

アプリケーションパッケージの業務ロジックは，仮想化環境上に構築した検証用の“簡易アプリケーションサーバ”の下で，Webベース・アプリケーションとして動作する。
簡易アプリケーションサーバは，該当する業務アプリケーションを呼び出し，処理を実行後，処理結果をクライアントパソコンへ返却する。

*三菱電機インフォメーションシステムズ㈱

1. まえがき

三菱電機㈱及び三菱電機インフォメーションシステムズ㈱(MDIS)では，様々な用途・業種に向けて業務アプリケーションパッケージを開発し，それを活用したビジネスを展開している。

本稿では，近年の技術動向及び市場動向を踏まえ，これらの業務アプリケーションパッケージをSaaS向けのシステムとして実現するための方式を，小売店向けの販売管理パッケージで検証した内容について述べる。

この方式は，仮想化された環境下に，業務アプリケーションパッケージに共通なアプリケーションサーバ機能を構築することで，既存のアプリケーションパッケージへの改修を最小に抑えるようにしているところに特長がある。この方式は，アプリケーションサーバ上で動作する多くの既存パッケージに適用できるものである。

今回，その実現性を検証するために，検証用の“簡易な”アプリケーションサーバ機能(以下“簡易アプリケーションサーバ”という。)を構築し，IaaSベンダーが提供する仮想化環境下での動作確認を行った。

2. SaaS化に向けた課題と指針

既存パッケージのSaaS化に当たっては，新しい実行環境への移植を可能な限り効率的に実現する必要がある。

このためには，移植対象となるパッケージ・アプリケーションへの改修を最小に抑え，現行のパッケージで実現している機能や操作性を損なうことなく実行できるようにすることが最も重要な課題となる。

インターネットを介して動作する既存の業務アプリケーションパッケージの大半は，アプリケーションサーバ上で業務ロジックを実行する構造となっており，SaaS化に際してもこの論理的構造を踏襲できる。しかしながら，アプリケーションサーバの仕様はベンダーごとに異なるほか，提供されるバージョンによっても異なり，その都度パッケージ側の改修が必要となる。

このため，個々の既存パッケージごとに，IaaSなどのベンダー環境に移植するための改修が必要となり，SaaS化するためだけに多くのコストと工期をかけることになる。

また，専用のネットワークで実行されていた業務アプリケーションをSaaS化する場合には，このほかにも①信頼性・可用性の確保，②セキュリティを中心としたデータベースの管理方法，③応答性能の確保などの技術課題のほか，④課金方法，⑤SLA(Service Level Agreement)など，多くの課題が存在するが，まずは，この移植効率の問題を解決する必要がある。

今回の検討では，この問題に対処するための方法として，アプリケーションサーバ機能の共通化と仮想化を採用し，次の点を検討指針とした。

(1) アプリケーションに手を加えない

今回検証した小売店向け販売管理システムなどの業務アプリケーションパッケージは，これまでの豊富な経験から，多くのノウハウと資産を継承している。

これらの資産を損なうことのないよう，アプリケーション(業務ロジック)には一切手を加えずに動作をさせる。

(2) 快適な操作性と応答性能の継承

中核となる基本入力機能は，応答性能及び操作性の良さを実現した既存パッケージと同等レベルに維持することが絶対条件となる。

(3) 信頼できるシステムの提供

インターネットを前提とした業務システムの特性上，セキュリティや可用性の問題が，関連する業務に甚大な支障を及ぼす可能性が高く，セキュアで高信頼なシステムであることが不可欠となる。

(4) アプリケーションとサーバの独立性の保持

アプリケーション又はアプリケーションサーバのいずれか一方に大幅な改修を実施したとしても，可能な限り他方は影響を受けないようにする。

(5) 他のアプリケーションパッケージへの適用

検証対象とした小売店向け販売管理パッケージだけではなく，将来的に他のアプリケーションへ適用することを見据えて，多種多様な環境での動作を目指す。

これらの点を踏まえて，その実現性を検証するための“簡易アプリケーションサーバ”を構築し，IaaSを提供するベンダーの仮想化環境下で動作検証を実施することとした。

3. 簡易アプリケーションサーバの構築

3.1 検証対象パッケージの基本アーキテクチャ

検証対象とした小売店向けの販売管理パッケージは，アプリケーションサーバにWebSphere(注1)(以下“WAS”という。)を使用することを前提としている。

WASは，JavaEE(Java Platform, Enterprise Edition)(注2)の仕様に準拠したアプリケーションサーバであるが，検証対象パッケージでは，業務ロジックを除き，ソースを自動生成するため，必要とするJavaEE API(Application Programming Interface)は極めて少ない。

3.2 簡易アプリケーションサーバの概要

実装は，今回の検証対象パッケージに必要な機能に限定した(**表1**)。サーブレットの実行など，必要最低限の機能は提供する一方，HTTPS(Hyper Text Transfer Protocol Security)，JSP(JavaServer Pages)(注3)タグへの対応など，不要な機能は一切実装していない。

(注1) WebSphereは，IBM Corp.の登録商標である。
(注2) Javaは，Sun Microsystems, Inc. の登録商標である。
(注3) JSPとJava Serverは，Sun Microsystems, Inc. の登録商標である。

3.3 簡易アプリケーションサーバのアーキテクチャ

簡易アプリケーションサーバ上でのアプリケーションソフトの動作イメージを**図1**に示す。

基本的な動作原理はJavaEEに準拠しており，クライアントアプリケーションからのリクエスト要求に応じたサーブレットを呼び出す。

サーブレット以降の処理は，パッケージ側のフレームワークが担当し，要求された業務ロジックを実行する。

処理結果は，JSPで生成したXML(eXtensible Markup Language)でクライアントアプリケーションに返送される。JSPの動作は，Javaソースを動的に生成し，コンパイルしたクラスファイルを実行することで実現した。

今回の検証対象パッケージでは，データの更新処理にEJB(Enterprise JavaBeans(注4))を採用しているが，このEJBは，検証対象パッケージ独自のフレームワークによって制御されている。このため，簡易アプリケーションサーバでは，EJBの実行に関する機能を実装していない。

これを補うため，EJBを制御しているフレームワークに対して，簡易アプリケーションサーバ上で実行する場合，EJBを通常のJavaBeansとして扱うようにフレームワークの改修を施した。

また，操作性を確保するため，1つのバッチファイルの起動，終了をするだけでサーバの起動，停止を実行できるようにし，バッチファイルを複数起動したり，コンソール管理画面からの操作などをしたりせずに済むようにした。

今回構築した簡易アプリケーションサーバでは，検証対象のパッケージに不要な処理はすべて実装を見送った結果，軽量・簡易な機能となり，仮想化環境にもかかわらず実用に足る応答性能を得ている。

なお，簡易アプリケーションサーバは100%Javaで記述されており，特定のOSやハードウェアに依存することなく多種多様な環境で稼働させることができる。

表1．簡易アプリケーションサーバのJavaEE対応状況

分類	機能	動作	備考
基本動作	アプリケーションの管理(EAR,WAR)	×	独自体系による，アプリケーションの配置と管理方式
	サーブレットの実行	○	URLに応じたサーブレットの実行が可能
	EJBの実行	×	実装せず。ただし，EJBを通常のJavaBeansとして扱うことで実行可能
	静的リソースの返却	○	画像ファイル，テキストファイルなどを返却可能。
	JSPの実行	△	ソース自動生成ツールで出力されたJSPのみ実行可能。JSPタグ，一部ディレクティブへは未対応
	マルチスレッド実行	○	指定スレッド数にて同時実行可能。指定スレッド以上のリクエストは，キューイング
	HTTPセッション管理	×	実装せず
	コネクション・プール	△	実装せず。ただし，オープンソースなどによって代替可能
管理	web.xml定義	△	アプリケーションサーバの定義ファイルに，サーブレットURLなどを定義
	Webアプリケーション再起動	×	Webアプリケーション単位での再起動は不可。サーバの再起動が必要

○：基本的な動作は可能
△：一部動作は未実装。又は，代替手段によって動作可能
×：未実装。動作せず
EAR：Enterprise ARchive
WAR：Web ARchive
URL：Uniform Resource Locator

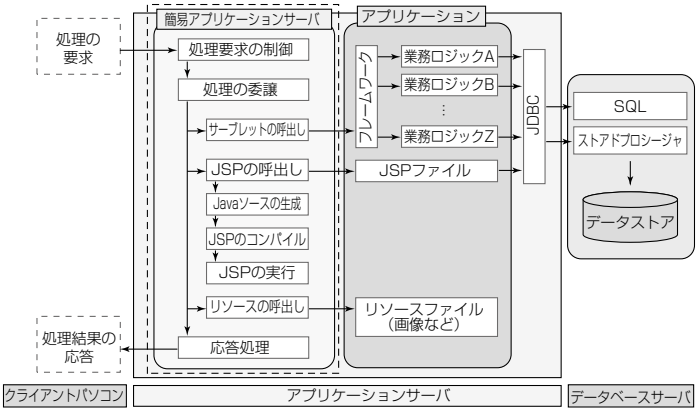


図1．アプリケーションソフトの動作イメージ

(注4) JavaBeansは，Sun Microsystems, Inc.の登録商標である。

4. 検　証

4.1 検証の方法

詳細な検証を行うためにVMware(注5)による仮想化環境を社内設備として構築し，開発した簡易アプリケーションサーバが，SaaS化に向けた指針を満たしているかどうかの検証を行った。

また，多種多様な環境での動作の検証を行うために，IaaSとして提供されている汎用的・一般的な商用プラットホームの下で検証を行うこととし，国内，海外を問わず幅広いベンダーからこれを選定して実施した。

(注5) VMwareは，VMware, Inc.の登録商標である。

4.2 検証の結果

(1) 仮想化環境での検証

検証用の仮想化環境では，簡易アプリケーションサーバ上で，対象アプリケーションが正常に動作することが確認できた。また，異なるバージョンのアプリケーション，並びにアプリケーションサーバを適用し，他方へ影響が発生せず，十分な独立性が確保できていることが検証できた。

操作性を確保するために必要な応答性能についても確認したが，従来のアプリケーションサーバ上での動作と比較すると，多少応答性能で劣ることが明確になった。これは，今回の製作で実装を見送った機能(コネクション・プール)の影響と考えられる。

(2) IaaS環境での検証

IaaS環境でも，簡易アプリケーションサーバ自体は問題なく動作した。ただし，帳票出力を行う場合，帳票サーバからプリンターへ印刷指示を行う方式については，方式の

変更が必要である。

現行の方式では，VPN(Virtual Private Network)等によって帳票サーバからプリンターを認識できる場合は印字可能であるが，パブリックなネットワークを介した場合，帳票サーバ側からプリンターが認識できないため，印字することができなくなるという問題がある。

4.3 検証課題への取組み

簡易アプリケーションサーバの動作検証に対しては，おおむね良好な結果が得られたが，一方で未実装の機能が原因で，期待した結果が得られなかった事項もある。

これらの事項への対処が今後の課題であり，仮想アプリケーションサーバとして実装が求められる次のような機能を中心に，この解決を図っていく。

(1) アプリケーションの管理／web.xml定義

アプリケーションサーバは，JavaEEに準拠した形式でアプリケーションの配置，管理を行っている。今回は，検証対象パッケージに都合の良い形式で管理したが，他のパッケージへの適用を考えた場合には変更が必要である。

今後，JavaEEに準拠した形式での管理体系に変更していく。

(2) EJBの実行

EJBは登場以来，まだ立場が曖昧(あいまい)な状況にある。最近のEJBは，かなり進化はしたものの，従来のJavaオブジェクトを動作させるなどの点で，否定的な面もうかがえる。また，EJBの動作理論上，EJBで実行した場合，通常のJavaオブジェクトより応答性能が低下するというデメリットもある。

EJBについては，先に述べた方式を取ることで業務ロジックは十分に動作するため，今後とも実装は不要と考えるが，更なる検証が必要である。

(3) HTTPのセッション管理

検証対象パッケージでは，リッチ・クライアント方式を採用しており，通信ごとに必要な情報を交換し合うため，HTTPセッションが不要であった。ただし，リッチ・クライアント方式ではない一般的なアプリケーションでは，サーバとのやりとりを頻繁に行う必要があるため，HTTPセッションが多く使われることになる。

今後，他のパッケージへの適用に向けて，HTTPセッション管理の実装を進めていく。

(4) コネクション・プール

データベースとのコネクションは，必要に応じて確立する方法を取ると，接続・切断のオーバーヘッドによって応答性能が劣化する。これを解消するために，多くのアプリケーションサーバでは，都度コネクションを切断せずに保持する機構(コネクション・プール)を備えている。

検証に使用した簡易アプリケーションサーバにはこの機能を実装しなかったが，実用化に当たっては，オープンソースであるTomcat(注6)アプリケーションサーバのコネクション・プール機構を実装する方向で検討を進めていく。

(5) Webアプリケーションの再起動

多くのアプリケーションサーバでは，サーバを再起動することなく，一部のアプリケーションのみを再起動(再読み込み)することが可能である。これは，Java VM(Virtual Machine)に用意されたクラスローダーをアプリケーション・サービスごとに用意することで実現されている。

今回の簡易アプリケーションサーバでは，この機能を実装していないため，同一アプリケーションサーバ上で複数のアプリケーション・サービスを提供した場合に，各サービス単位での保守がしにくくなる可能性がある。

(注6) Tomcatは，Apache Software Foundationの商標である。

5. むすび

今回の検証によって，アプリケーションサーバ上で動作する小売店向け販売管理パッケージのソフトウェアを，汎用的なIaaS環境下で動作させるための方法が確立できた。

これによって，このパッケージの機能をSaaSとして提供するための方法に技術的な目処(めど)が立ったほか，アプリケーションサーバのバージョンアップに伴うアプリケーションパッケージ側の改修を最小に抑えることについても大きく前進した。

このソリューション技術は，既存の業務アプリケーションパッケージのSaaS化に有効であるほか，業務アプリケーションパッケージの維持開発費を軽減する方法としても，十分な効果が期待できる。

今後は，優良なベンダーの選定やデータベースの管理をはじめとしたサービス化への課題にも取り組み，より利便性の高いサービスの実現を目指していく所存である。