サーバレスアーキテクチャーによる作業量削減効果の測定

－ビジネスアジリティは向上するのか？－

IA−001：サーバレスアーキテクチャーのビジネス適用：石野 大輔

How to write Paper

－　Format Template　－

Team’s name：Leader’s name

新たなデジタル技術が日々開発されている状況の中，企業は競争力強化のため，そのような新技術を活用してビジネスモデルを素早く創出することが求められている．近年，ビジネスアジリティを向上させるアーキテクチャとしてサーバレスアーキテクチャーが注目されている．本論文では，サーバレスアーキテクチャーの適用により，ビジネスアジリティが向上するのか検証した．検証で得られた結果から，ビジネスアジリティ向上の是非と今後の課題・展望について述べる．

This manual guides you in writing an “JGS Paper” in the proper format and structure. The guideline is prepared as an input format template to help you write a properly formatted paper by following the instructions explained in the body. The abstract of your paper should be written in this style, with five key words and phrases as shown below.

Key Words & Phrases ：執筆要領，論文テンプレート，発明提出，フォーマット，参考文献

　　　　　　　　　　 paper guideline, paper template, invention submission, format, references

１．はじめに

日々新たなデジタル技術が開発されている昨今，IT業界のみならず様々な業種でデジタル技術を駆使したビジネスが創出され，企業価値を高めている．こうした中で，各企業は競争力維持・強化のために，デジタルトランスフォーメーション（DX：Digital Transformation）をスピーディに進めていくことが求められている。我が国企業においては、多くの経営者が DX の必要性を認識し，DX を進めるべく，デジタル部門を設置する等の取組が見られる。しかしながら、PoCを繰り返す等、ある程度の投資は行われるものの実際のビジネス変革には繋がっていないという状況が多くの企業に見られる現状にある．既存システムが老朽化・複雑化・ブラックボックス化する中では，データを十分に活用しきれず，新しいデジタル技術を導入したとしても、データの利活用・連携が限定的であるため、その効果も限定的となってしまうという問題が指摘されている．経済産業省のレポートによれば，老朽化・複雑化した既存システムがこのまま継続して使用された場合，2025年～2030年の間に最大12兆円の経済損失が生じると推定されている[1]．

既存の IT システムを巡る問題を解消しない限りは，DXを本格的に展開することは困難であると考えられる．しかし既存のITシステムの改修は大規模かつ長期のプロジェクトとなるため，スピーディに効果を得ることができない．そこで，頻繁に改修が求められるビジネスロジック又は新規に作成するビジネスロジックについて，新たなデジタル技術を使用して構築することで，ビジネスモデルの変化に迅速に追従できるようにする必要がある．

近年，ビジネスアジリティを向上させるアーキテクチャとしてサーバレスアーキテクチャーが注目されている。サーバレスアーキテクチャーは，各クラウド事業者の責任範囲が広く，ユーザは業務ロジック構築とアプリケーション開発に注力できる点が大きな特徴である．本論文では，サーバレスアーキテクチャーの適用によりビジネスアジリティが向上するのか，モデルシステムを実装することで検証を行う．以下，2章でサーバレスアーキテクチャーの特徴を述べ，3章，4章では，成果物の作成量及び構築作業時間の2つの観点で，検証結果について述べる．また，5章，6章にて検証作業にて確認された課題と課題を踏まえた今後の展望を述べる．

２．サーバレスアーキテクチャーとは

　本論文では「サーバレス」，「サーバレスアーキテクチャー」の２つの用語は、2019年度JGS論文「サーバレスアーキテクチャーの適用検討プロセスの提案」[2]の定義と同意とする．

サーバレスとは、「サーバ管理を必要としないアプリケーションの構築と実行の概念」[2]のことであり、代表的な実現方法としてFaaS（Function as a Service）とBaaS（Backend as a Service）がある．これらFaaSとBaaSの個別機能を組み合わせたシステム構成をサーバレスアーキテクチャーとする．[2]

2.1 サーバレスの特徴

　サーバレスには, クラウド事業者ごとに様々な特徴が存在するが，ここでは代表的な特徴について述べる．

1. サーバの構築・運用が不要

サーバやランタイム等のアプリケーションの実行に必要な基盤環境については，図１のとおり、クラウド事業者の責任の下，運用・管理されている．そのため、ユーザーは環境構築を行う必要がなく、運用・管理に掛かる作業負荷も大幅に削減される．



図１．サーバレスの責任モデル

1. 柔軟なスケーラビリティ

処理量に応じて，自動的なスケーリングが実施可能．

３．サーバレスアーキテクチャー適用によるビジネスアジリティ向上の検証

本章では，1章で述べたサーバレスアーキテクチャーの適用効果を測定する．サーバレスアーキテクチャーを使用して東京オリンピック2020グッズ購買サイト(以下、グッズ購買サイト)を構築し，オンプレミス環境における構築作業と比較することで，ビジネスアジリティが向上することを検証した結果を述べる．

検証範囲は，意思決定された事業戦略を，IT部門がサーバレスアーキテクチャーを適用して開発する「システム開発部分」とする．

本論文におけるビジネスアジリティの向上とは，サービスリリースまでの開発期間が短縮されることで，迅速にサービス提供可能となることを指す．

　開発期間の短縮は，以下の観点で検証する．

・ドキュメント作成量

・構築作業時間

3.1仮説と前提

3.1.1.仮設

サーバレスアーキテクチャーの適用により，各開発工程のドキュメント作成量の減少，構築作業時間の短縮により，開発期間が短縮することを仮設とする．　#直前の一文に仮説がかいてある

3.1.2.前提

1. 利用するクラウドベンダー

本検証では，代表的なクラウド事業者[3]であるAmazon Web Serviceが提供するサーバレスサービスを使用して，グッズ購買サイトのシステム構築を行う．

1. モデルシステム選択理由

　政府がキャッシュレス決済を推進しており，ECサイトの立ち上げが増加すると考えられるため，グッズ購買サイトに決定した．

増加の根拠について述べる．経済産業省の資料では，キャッシュレス決済推進のメリットとして「店舗の効率化・売上拡大」，「データ利活用」が挙げられている．[4]実店舗で商品を販売する企業が，ECサイトを立ち上げることを想定した場合，販売チャネルの多角化による売上の拡大が期待でき，サイト内の購買情報をデータ収集し、分析・活用することで，マーケティングや商品開発に繋げることが可能になる．ECサイトはキャッシュレス決済が主な決済手段であり，キャッシュレス決済推進によるメリットを享受できることから，ECサイトが増加すると考える．

1. グッズ購買サイト構築

グッズ購買サイトに必要な要求事項を要求仕様書（別冊A-1）として纏め，要求を充足するシステムを構築する．適用するサーバレスアーキテクチャーは，AWSにより品質が保証されているAWS samples掲載のアーキテクチャ設計より，検証要件に類似した要件のServerless Airline Booking（航空券購買システム例）[4]を参考に検討を行った．システム構成は図2に示す．

スクリーンショットの画面

自動的に生成された説明

図2 グッズ購買サイトのシステム構成

3.2検証方法

3.2.1構築作業時間　#ガイドに出典いるか？

手順A-1：要求仕様書を基に，採用するAWSサービスで必要な設計や設定について，AWS開発者ガイドやガイド記載のチュートリアルに取り組むことで確認する．

手順A-2：確認結果を整理し，システムで扱うデータ定義をデータ定義書（別紙B-1）として纏める．AWSコンソール作業で設定する値を，グッズ購買サイトで実装する機能ごとに，パラメータシート（別紙B-2）として纏める．AWS CLIで必要となる手順をCLI作業手順書（別紙B-3）として纏める．

手順A-3：AWS Lambdaがサポートするランタイムを利用して，グッズ購買システムのアプリケーションソースコードを開発する．コード開発は，メンバーで分担して開発する．

手順A-4：手順1で使用した開発者ガイドやチュートリアルの手順をベースに，準備した定義書，パラメータシート，作業手順書に従い，AWSコンソール，AWS CLIを利用してグッズ購買サイトを構築し，作業時間を計測する．構築作業は，AppSyncクエリからStep functionsのステートマシンを起動するのに必要な認証機能の実装を除き，AWSを利用した開発未経験者であるメンバー1名が担当する．

3.2.2ドキュメント作成量

手順B-1：オンプレミス環境でウォーターフォールモデルによる開発を行う場合に必要となる成果物一覧を作成する．

手順B-2：3.2.1構築作業結果より，サーバレスアーキテクチャー適用による開発で必要となる成果物を検討し，作成済みの成果物一覧に記載する．

手順B-3：オンプレミス開発におけるドキュメント作成時の作業ボリュームを基準として、同じドキュメントをサーバレスアーキテクチャーの適用による開発を行った場合と比較，評価する．評価は以下の3段階とする．

・作業ボリュームが同じ

・作業ボリュームが減少する

・ドキュメント作成が不要となる

手順B-4：手順B-3にて「作業ボリュームが減少する」，「ドキュメント作成が不要」と評価したドキュメントついて，サーバレスアーキテクチャー適用時のメリットとして根拠を記載する．

上述の手順B-1～手順B-4を実施した結果を評価シート（別紙C-1）として纏める。評価シートには，3.2.1構築作業時に顕在化した課題についても記載する．

3.3検証結果

3.3.1 構築作業時間

グッズ購買システムの構築作業時間（3.2.1の手順A-3，手順A-4の合計）は10時間であった．内訳は，手順A-3：アプリケーションソースコードの開発時間がのべ7時間，手順A-4：AWSコンソール，AWS CLIを利用した構築作業時間が3時間であった．

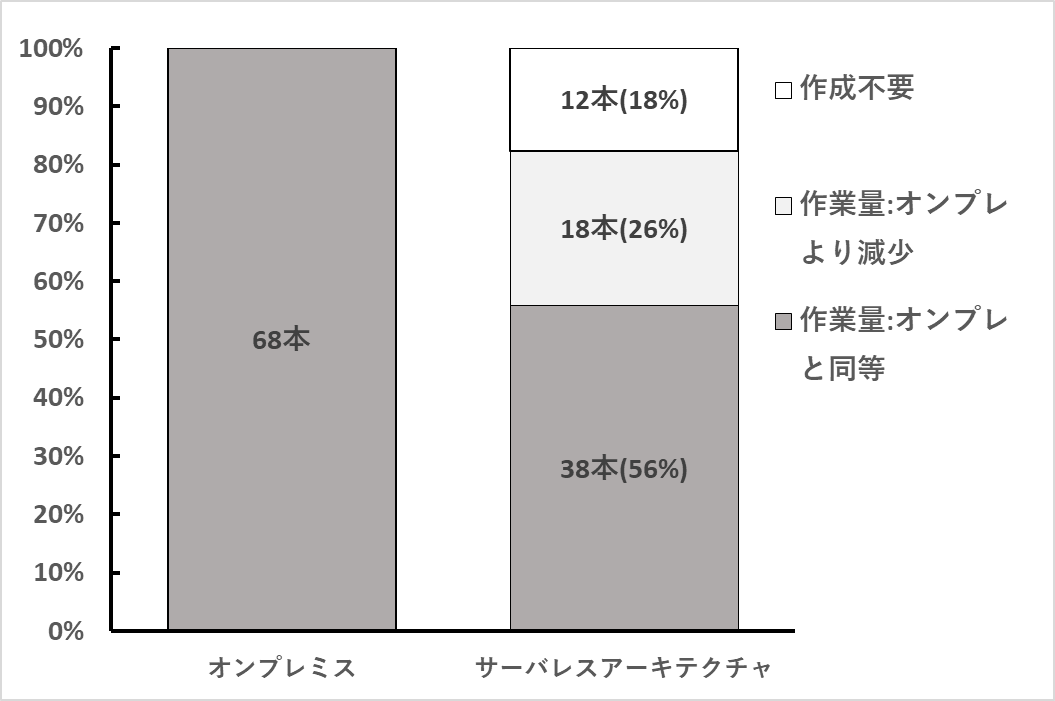


図3.ドキュメント作成量検証結果

3.3.2ドキュメント作成量

　ドキュメント作成量について検証した結果を図3に示す．

　オンプレミス環境でウォーターフォールモデルによる開発を行う場合に必要なドキュメントは68本，サーバレスアーキテクチャーの適用による開発で必要なドキュメントは56本であった．サーバレスアーキテクチャーの開発ではオンプレミス開発との比較でドキュメント作成量が18%減少した．

オンプレミス開発におけるドキュメント作成時のボリュームを基準として、同じドキュメントをサーバレスアーキテクチャーの適用による開発を行った場合の作業ボリュームについて，以下に記載する。

・オンプレミス開発時と作業ボリュームが変わらないドキュメント：38本

・オンプレミス開発時より作業ボリュームが減少するドキュメント：18本（サーバレスアーキテクチャーの適用による開発全体の32%）

４．考察

3.3 検証結果より，サーバレスアーキテクチャーを適用した開発では，非機能部分のドキュメント作成量および構築作業時間が削減されることを確認できた．まず，「非機能部分」がサーバレスだとどのように減るのか述べる．

4.1 非機能部分の作業がサーバレスで減る理由

　クラウドサービスの利用形態と図1.サーバレスの責任モデルを用いて説明する．

・IaaS（Infrastructure as a Service）

「仮想化」以下のインフラ部分は，クラウド事業者により高度に抽象化されており，ユーザーはインフラの導入・構築、管理が大幅に省力化される特徴がある．

・PaaS（Platform as a Service）

IaaSの特徴に加え，「ミドルウェア・ランタイム」が提供される．ベンダーが提供する仕様に依存するため，自由度や柔軟性に制限を受けるが，各種ミドルウェア・ランタイム導入・設定が省力化される特徴がある．また，拡張性、可用性を向上させる機能が提供されており, 設定も可能である．

・サーバレス

上述のIaaSやPaaSの特徴に加え，拡張性，可用性を向上させる機能がネイティブとして組み込まれている。ユーザーの設定に対する責任が低いため，設計量（ドキュメント作成量），構築作業量が削減される．

構築で利用したサービスを例に，拡張性・可用性が向上させる機能が組み込まれていることを確認する．

(1)DynamoDB

・拡張性

DynamoDBはキャパシティユニットを増減させることで処理性能が決まる．キャパシティユニットとは，1秒あたりにN回の読み書きできる容量を指し，読み込み／書き込みの単位に設定可能である。ユーザーはシステム要件に合わせて適切に値を設定する，もしくはAuto Scalingを利用可能である．

・可用性

デフォルトで3つのアベイラビリティゾーンにデータが分散される．DynamoDBグローバルテーブル（マルチリージョンにマルチデータベースをデプロイ）の場合，サービスレベルアグリーメント（SLA）は99.999%以上となる．

(2)AWS Lambda

・拡張性

　Lambda関数を処理するインスタンスは自動でスケーリングされ，並列で処理される．リージョンごとに，1000件（東京リージョン利用時）まで同時実行を処理する．

・可用性

　Lambdaは複数のアペイラビリティゾーンで関数を実行し，１つのゾーンでサービスの中断が発生した場合にも，Lambda関数をイベントの処理に使用できることを保証されている．

上記からも，サーバレスには拡張性，可用性を向上させる機能がネイティブとして組み込まれていると言える．

4.2 構築作業時間

　構築作業時間が大きく短縮される理由について，手順A-3、手順A-4ごとに考察する．

4.2.1 手順A-3：アプリケーションソースコードの開発時間が短くなる理由

　直接要因は，各Lambda関数が「目的が1つもしくは2つで，ステートレス」な設計となっていることである．関数で実現することが簡素なので，プログラムステップ数が少なく，テストをし易くなる．ステートレスなので，セッション情報の管理を考慮する必要も無くなる．

　根本要因は，選択したサーバレスアーキテクチャーは，サービス間の結合度が低く，他サービスへの影響を考慮する必要が無かったことにあると考える．

4.2.2 手順A-4 AWSコンソール，AWS CLIを利用した構築作業時間が短い理由

　サーバレスの各サービスは，クラウド事業者責任範囲が広いため，ユーザーが設定値を入力して構成を確定（保存）すると，自動でサービスが構築されるため．

4.3 ドキュメント作成量

　サーバレスアーキテクチャーを適用した場合に，「作成が不要」となるドキュメントは，非機能の中でもIaaSに該当する部分である．

オンプレミス環境で開発時よりドキュメントの作業ボリュームが減少する理由としては，ユーザーは要求事項に合わせて，サーバレスの各サービスが提供する機能（バックエンド接続，拡張性，バックアップ等）を利用することを選択し，設定値を決めれば良いためである．構築作業は、ユーザーが設定値を入力して構成を確定（保存）すると，クラウド事業者にて自動構築される．

4.4. 結論

　サーバレスアーキテクチャーを適用することで，サービスリリースまでの開発期間が短縮され，迅速にサービス提供可能になる．結果、ビジネスアジリティが向上すると言える．

4.5 サーバレスアーキテクチャー適用により作業量が減少しない作業について

　本検証では，業務ロジックおよびアプリケーション開発に対しては，開発期間の短縮効果は確認できなかった．2章で述べたとおり，サーバレスとは「サーバ管理を必要としないアプリケーションの構築と実行の概念」である．業務要件定義「対象業務の現状を分析し，新たに実現すべき業務の流れを明確化する」や，アプリケーションの構造・設計、利用するデータの設計はユーザーの責任範囲である．

~~3.3検証結果より，サーバレスアーキテクチャーを適用することで開発期間の短縮が可能であり，ビジネスアジリティの向上が確認できた．開発期間が短縮された主たる要因は， [参照]評価シートより，非機能関連のドキュメント作成量および構築作業時間が大きく削減されたことである．~~

~~一方で，本検証では，業務ロジックおよびアプリケーション開発に対しては，開発期間の短縮効果は確認できなかった．~~

５．サーバレスアーキテクチャーの適用おける課題

今回の検証結果より、次のような課題が認められた。

1. 開発言語の選択

Lambda関数構築時に、開発者の得意な言語をそれぞれの関数で選定したため、複数言語が混在したシステムとなった。

AWSのLambdaでは、複数の言語をサポートしているため、開発工程前にコーディング規約を定める必要がある。

1. 運用・監視の複雑化

検証作業にて、エラー原因を特定するため、ログの解析を行ったが、当初ログの設定が無効になっており、問題切り分けに難航した。

また、ログの出力先や分析方法が各サービスで異なる為、各々のサービスの連携に関してエラー分析が複雑化し、従来システムのような統一的な運用が難しい。

1. クラウド人材の育成

AWS未経験者にて本検証を行ったため、構築作業前の事前学習に延べ１ヶ月程度費やした（AWSの基本概念、各サービス仕様等）。AWSサービスは日進月歩で進化しており、エンタープライズレベルの開発を行うためには、クラウド人材の育成が必須となる。

６．今後の展望

サーバレスアーキテクチャーを用いることで，開発時に作成するドキュメント量と開発作業量が削減され，ビジネスアジリティが向上することが確認された．加えて，今回検証できていない以下の２つの観点を追求することで，更にビジネスアジリティを向上させる可能性があると考える.

1つ目は，開発業務への注力である．サーバレスアーキテクチャーを適用し運用業務を省力化することで，運用に割いていたコスト・人員を開発業務に充てられるため，よりスピーディな開発が可能になると考える．

２つ目は、各サービスの疎結合化である.各アプリケーションをマイクロサービスとして独立させ，機能追加・仕様変更時の際も対応範囲を局所化することで，低コスト・短納期で開発が可能であると考える.また上記の観点に加え，ドメイン駆動設計やアジャイル開発を組み合わせることで，より大きな効果が期待される.

ただし，これらを実現させるためには、開発と運用の一体的体制やアジャイル開発を推進する組織風土の醸成に加え、高度な技術力とクラウドに対する豊富な知識を持つ人材の育成が必要である．

参考文献　　　　(記述例，[3]は架空のもの)

1. 大野侚郎，「論文作法」基礎の基礎，2002年度IBMプロフェッショナル論文発表会講演社内限定資料，2002年11月
2. 木下是雄，理科系の作文技術，中公新書，ISBN4-12-100624-0，1981
3. Gartner,Magic Quadrant for Cloud Infrastructure as a Service, Worldwide, Published 16 July 2019
4. Serverless Airline Booking, https://github.com/aws-samples/aws-serverless-airline-booking,2020.6.12
5. IBM Japan, http://www.ibm.com/jp/, 2003.2.28

添付１：メンバーリスト ( 会社名アイウエオ順 )

添付２：Member list

(alphabetical order of company name )添付１：メンバーリスト ( 会社名アイウエオ順 )

添付２：Member list

(alphabetical order of company name )

