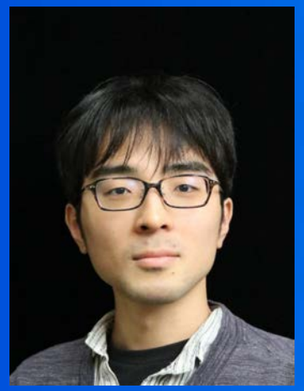
aws INNOVATE
ONLINE CONFERENCE

アクセスの急増に耐え得るサーバーレスアプリケーションを作る方法

木村 秀平
アマゾン ウェブ サービス ジャパン
技術統括本部 ソリューションアーキテクト



Twitterハッシュタグ #AWSInnovate

本セッションについて

セッションの対象者

- アクセスの急増が予測されるアプリケーションで、サーバーレスのメリットを活かしたい方
- AWS Lambdaを使用したことがある方

セッションのゴール

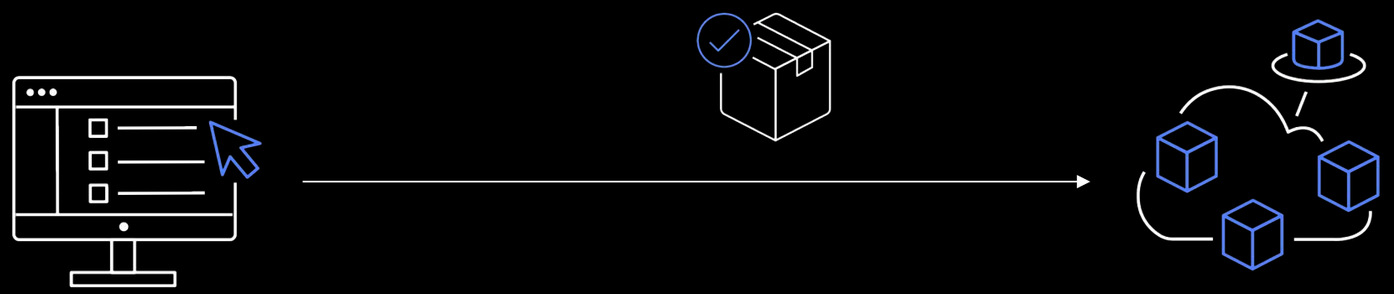
- アクセスの急増に耐え得るサーバーレスアプリケーションを作るために必要なことを理解する
- AWS Lambdaのスパイク時に注意すべき仕様を理解する

今回扱う課題リスト

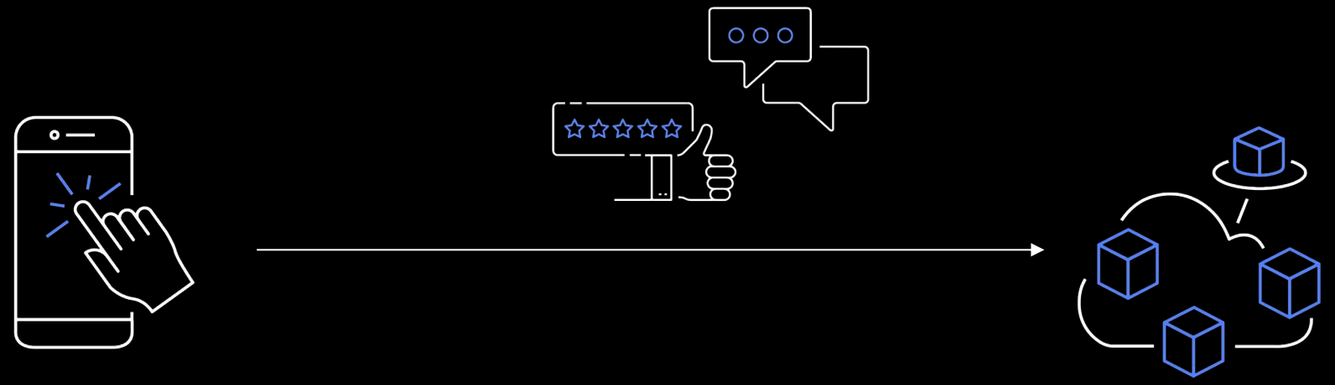
- AWS Lambdaの同時実行数制限とコールドスタートの解消を行う

【ご連絡事項】
本セッション内容について確認するためのクイズおよび簡単なアンケートがセッションの最後にあります。
なおクイズの回答はアンケートとあわせて表示されます。

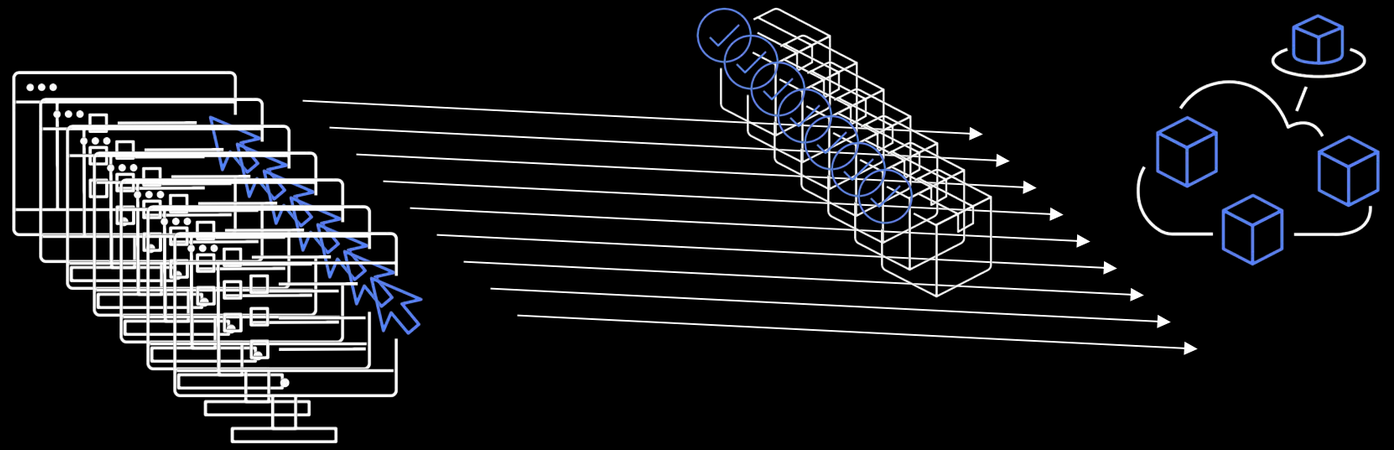
ECサイト



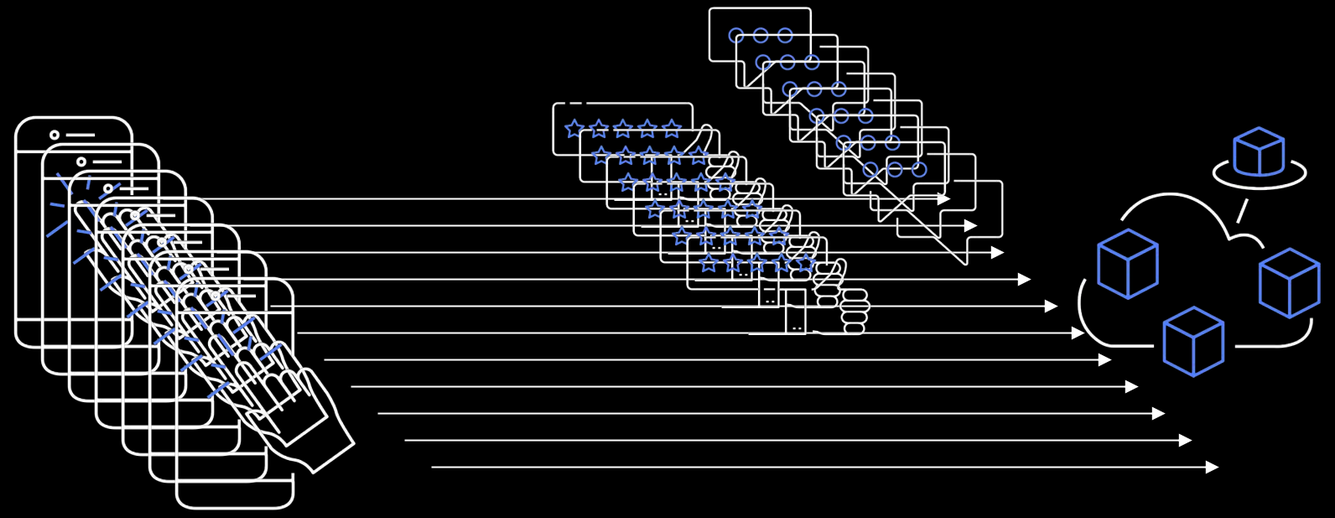
ゲーム



ECサイト
（Black Friday）



ゲーム
(リリース直後)



アジェンダ

アクセス急増（スパイク）の課題

サーバーレスサービスの利点と活用

Lambdaの注意点

スパイクに対処する

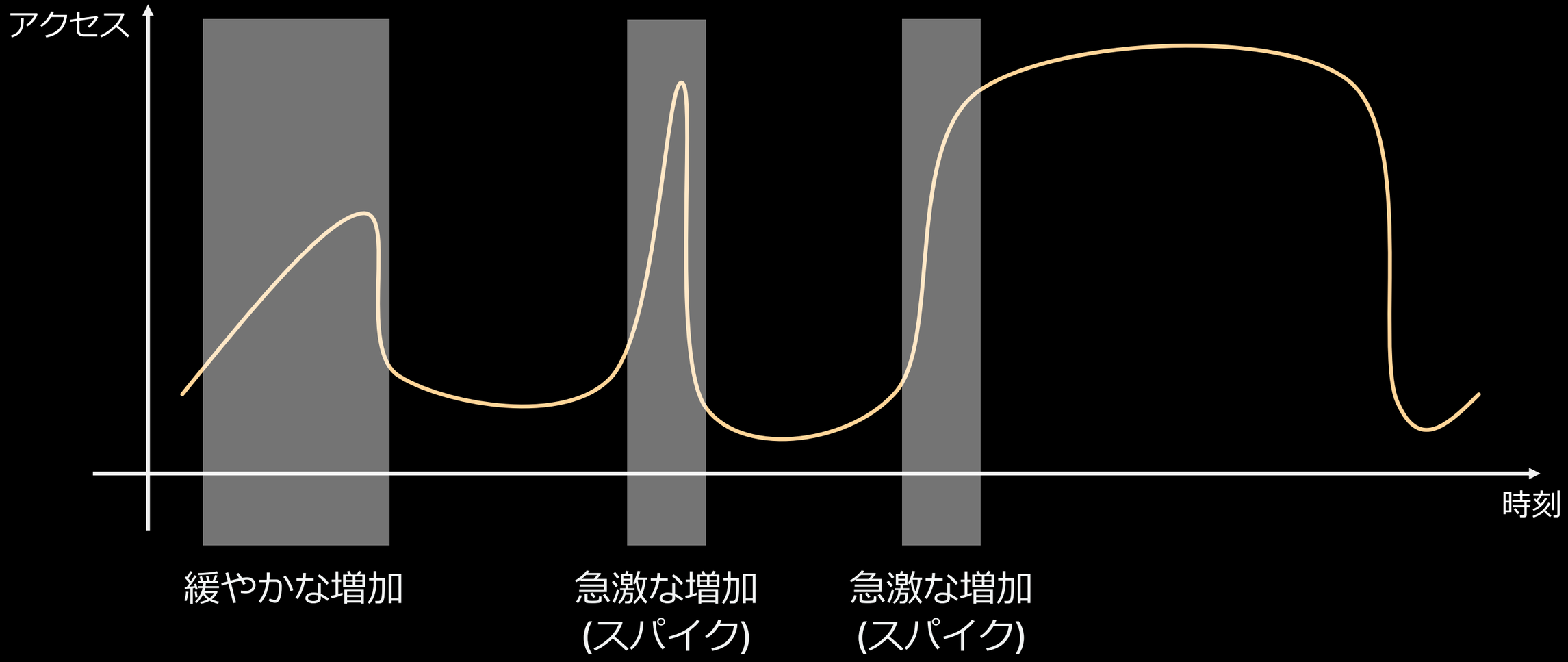
まとめ

クイズ

アクセス急増（スパイク）の課題

aws INNOVATE
ONLINE CONFERENCE

アクセスの急増（スパイク）



スパイク時に発生する問題

スケールアウトするアーキテクチャだったが、

- ECサイトでセール開始時に**アクセスが急増し、サイトが落ち**てしまい売上が伸びなかった
- モバイルゲームの**リリース直後にプレイヤーが殺到し、メンテナンス**に入る必要が生じユーザーをロストした

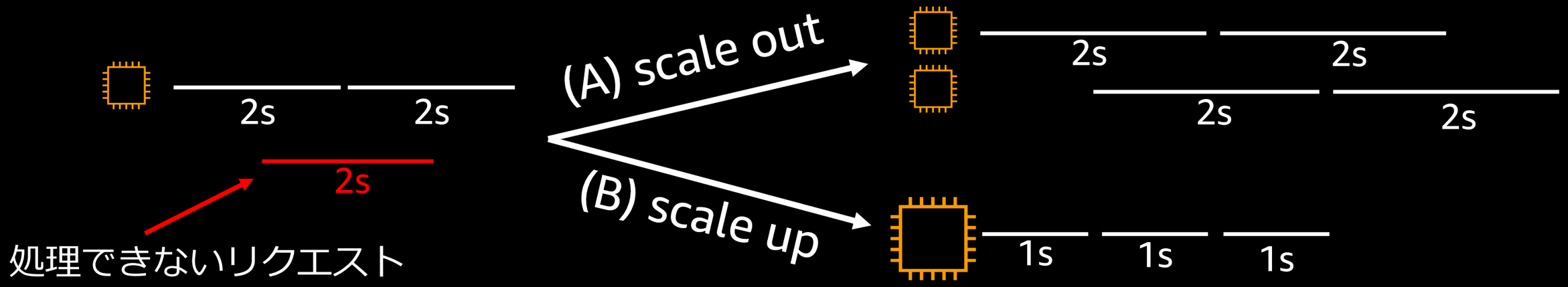
→ **スケールアウトには時間が必要**なため、スパイク時に必要なリソースが確保できず**ビジネス的損失**が発生し得る

スパイクを乗り越えるには

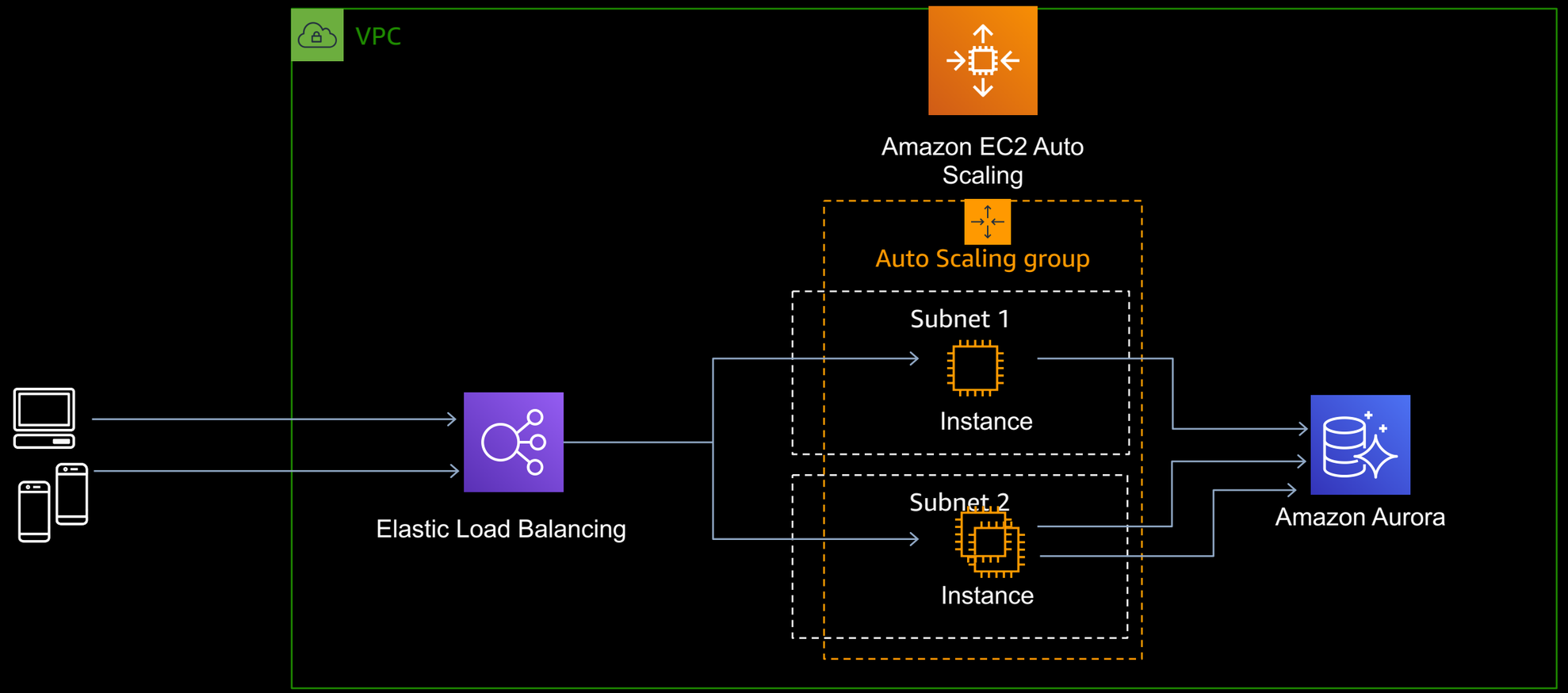
A. 大量の同時アクセスを捌く同時実行数（インスタンス数。水平スケーリング)
B. 実行速度の向上によるスループット増加（スペック。垂直スケーリング）

例えば、1秒ごとに1のアクセスがある場合

- 1の同時実行が可能なら、処理時間が**1秒**以下でないと全て捌けない
- 2の同時実行が可能なら、処理時間が**2秒**以下でないと全て捌けない

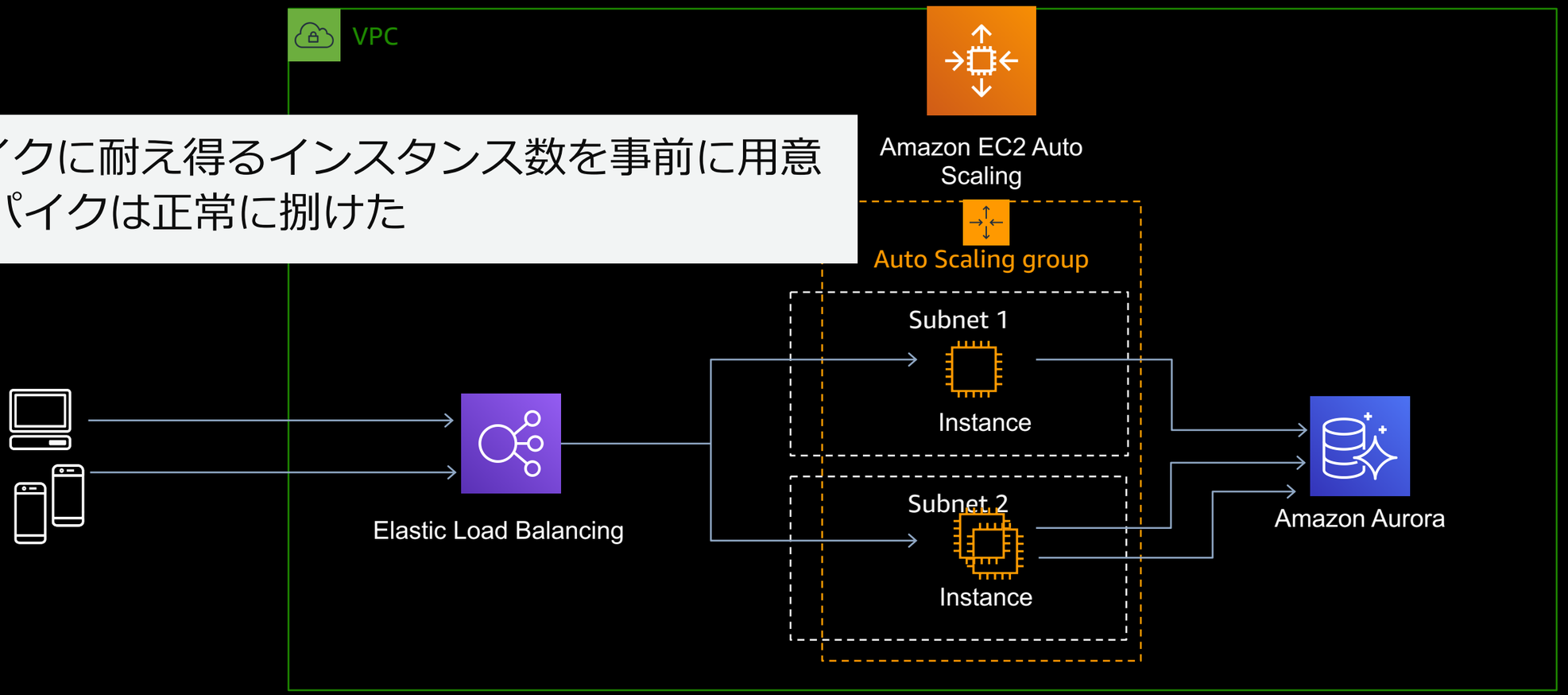


シンプルなAPIサーバー（EC2版）



シンプルなAPIサーバー（EC2版）

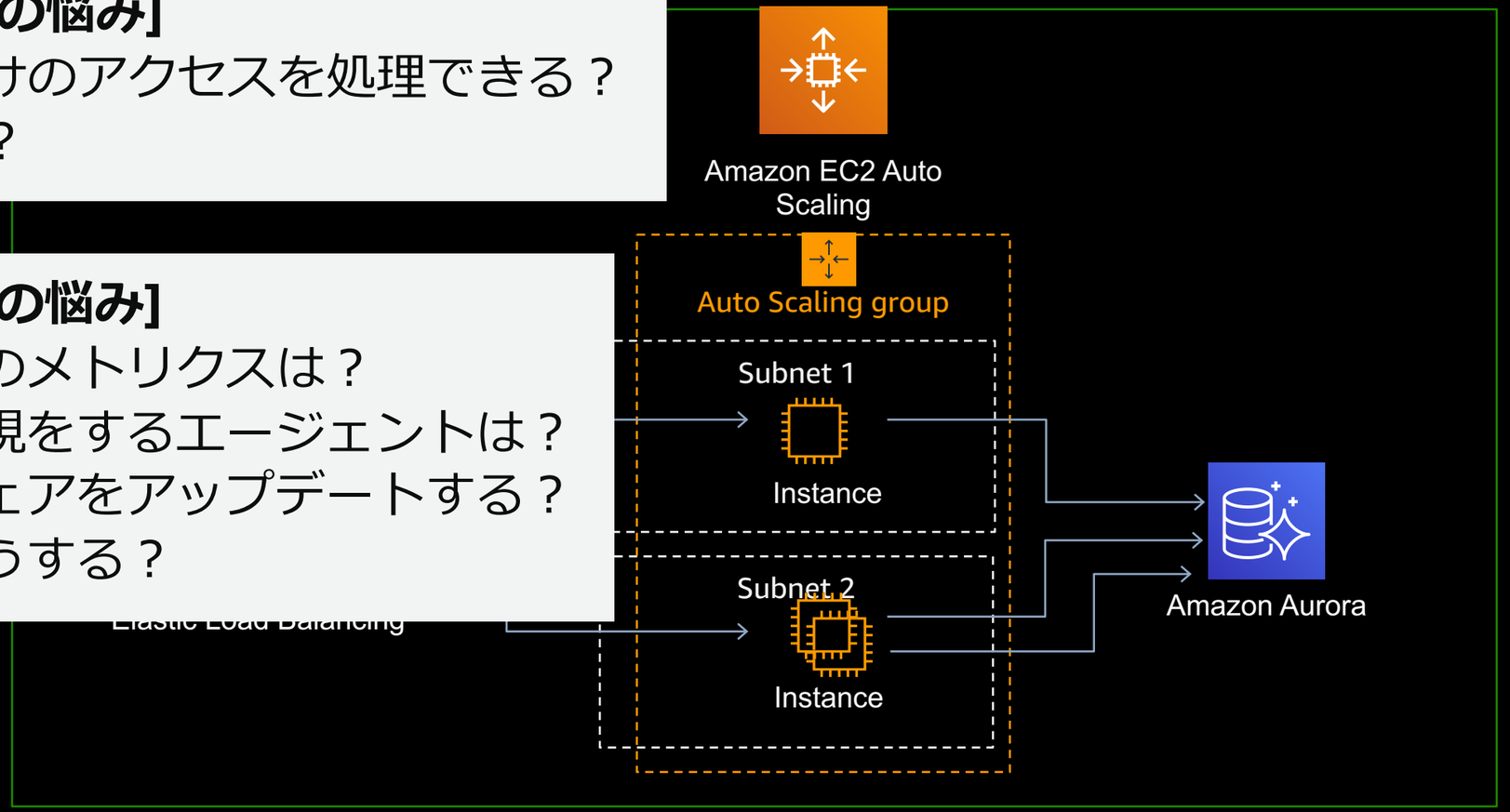
スパイクに耐え得るインスタンス数を事前に用意
→ スパイクは正常に捌けた



シンプルなAPIサーバー（EC2版）

[スケーリングの悩み]
一台でどれだけのアクセスを処理できる？
何台用意する？

[他にも、運用の悩み]
スケーリングのメトリクスは？
ログ収集や監視をするエージェントは？
どうミドルウェアをアップデートする？
デプロイはどうする？



スパイクするアプリケーションを作る際の期待

1. 運用負荷を減らしたい

2. スパイクに対応したい
    1. 同時実行数を増やしたい
    2. アクセスごとの実行時間を減らしたい

サーバーレスで構築しつつスパイクに対応したい

サーバーレスサービスの利点と活用

aws INNOVATE
ONLINE CONFERENCE

（おさらい）AWS サーバーレスサービスの利点

運用・管理負荷の低減

- インスタンスの管理が不要
- AWSのロギング・監視機能など他サービスとの統合
- 高可用性と耐障害性

最適なリソースの割り当て

- 柔軟なスケーリング
- 使った価値だけの支払い

セキュリティ

- OSのセキュリティもAWSが担当。アプリケーションや設定上のセキュリティに注力できる

https://aws.amazon.com/jp/serverless/

（おさらい）AWSが提供するサーバーレスサービス（一部）


AWS CloudWatch


Amazon Kinesis


Amazon DynamoDB


Amazon EventBridge


AWS Step Functions


AWS CloudTrail


Amazon SES


AWS Lambda


Amazon API Gateway


AWS IoT Core


Amazon SQS


Amazon SNS


Amazon Route53


Amazon Cognito


Amazon Pinpoint

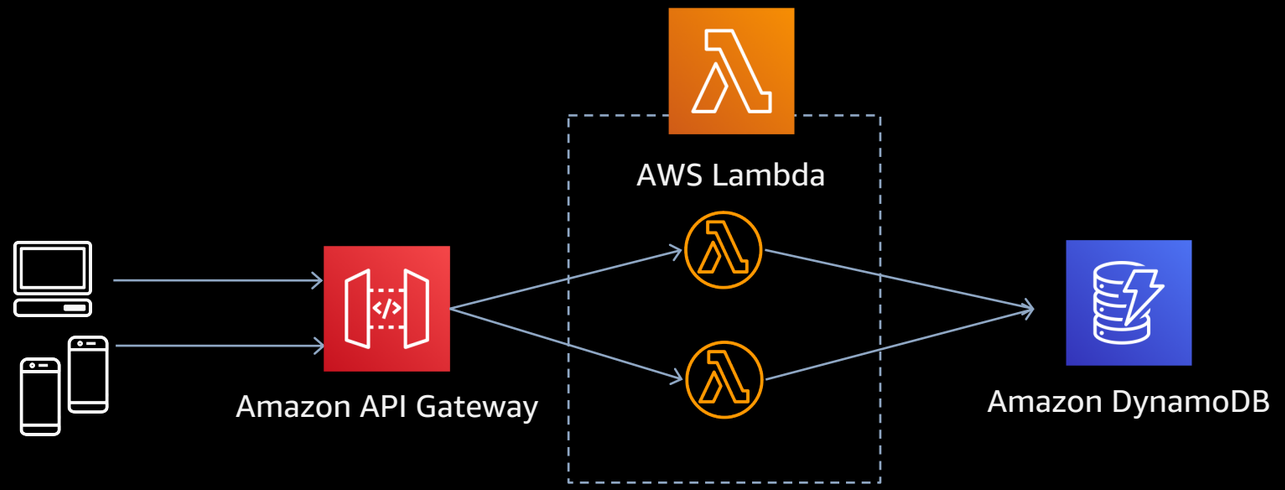

Amazon S3


Amazon Athena


Amazon Elastic Transcoder

シンプルなAPIサーバー（サーバーレス版）

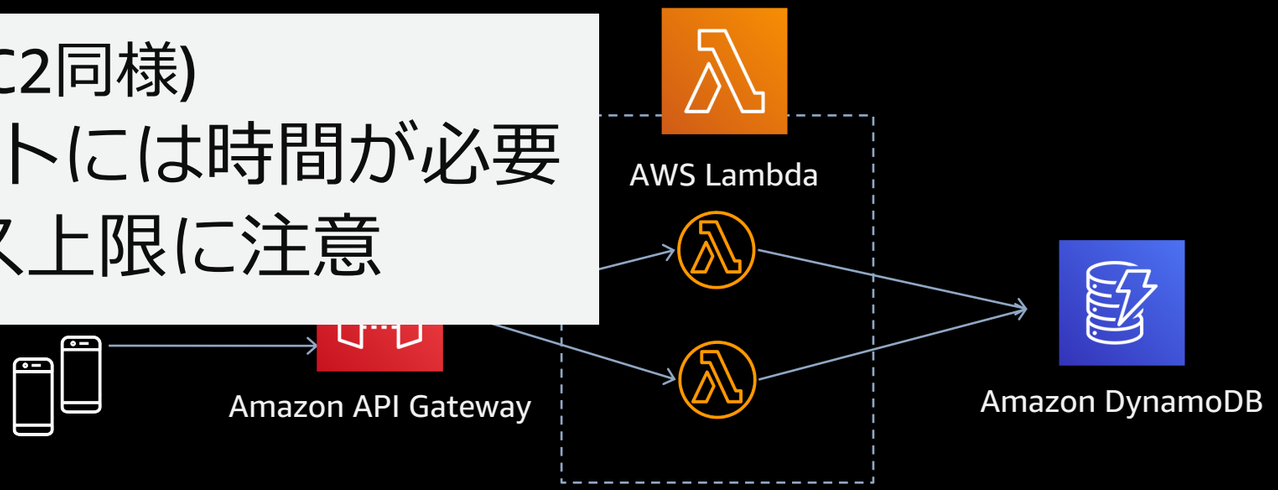
API Gateway+Lambda+DynamoDBで置き換える



データモデルの変更が困難でRDBMSはそのまま使いたい、というパターン
→ 後ほどご紹介だけ

シンプルなAPIサーバー（サーバーレス版）

(EC2同様)
スケールアウトには時間が必要
サービス上限に注意



Lambdaの注意点

aws INNOVATE
ONLINE CONFERENCE

Lambdaの仕様

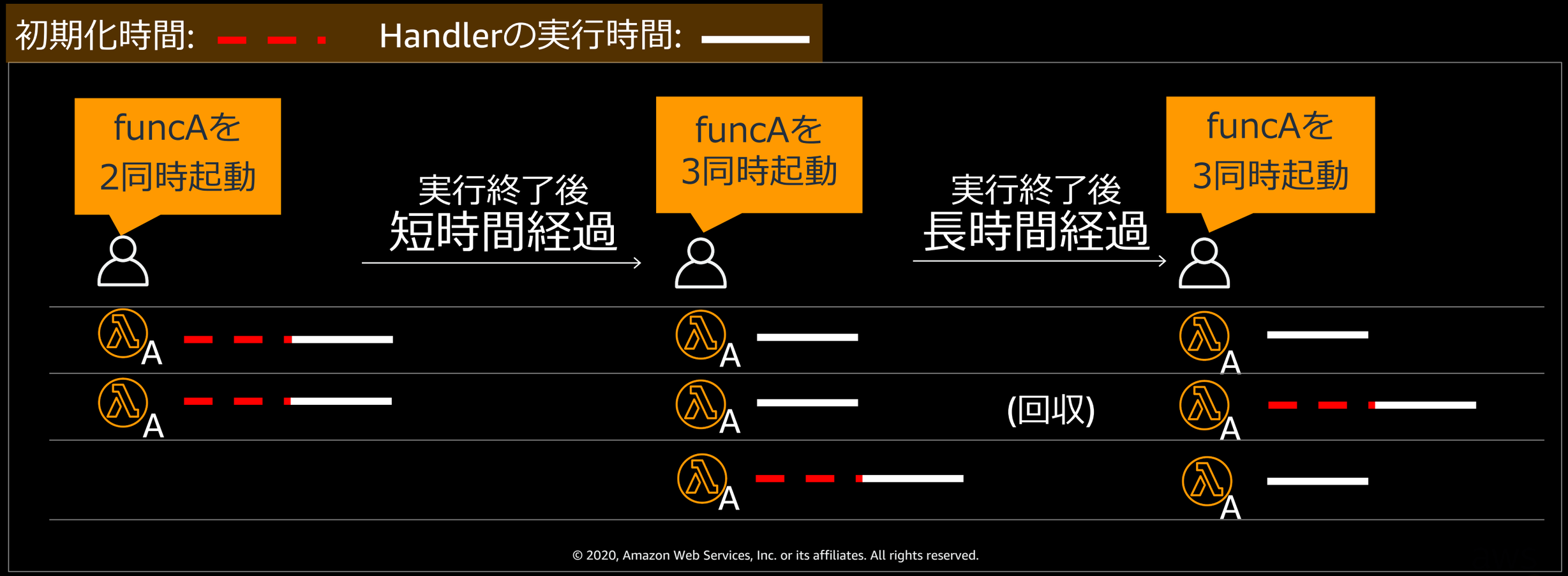
- 関数インスタンス（=Lambdaの実行基盤）で初回実行の場合、初期化が必要
- 同時実行数の上限はデフォルトで1,000（アカウント・リージョン単位）
- 同時実行数が足りない場合はThrottlingエラーが発生

- 最長実行時間は15分
- メモリは128MB-3,008MB
  - vCPUはメモリに比例して向上
- 状態は環境変数か外部サービスから取得
- Lambda内部で書き込み可能な領域は /tmp 以下の512MB

Lambdaの初期化（cold start）

関数インスタンスで初回実行の場合、初期化が必要

- 既存の関数インスタンスが回収されるタイミングは指定できない



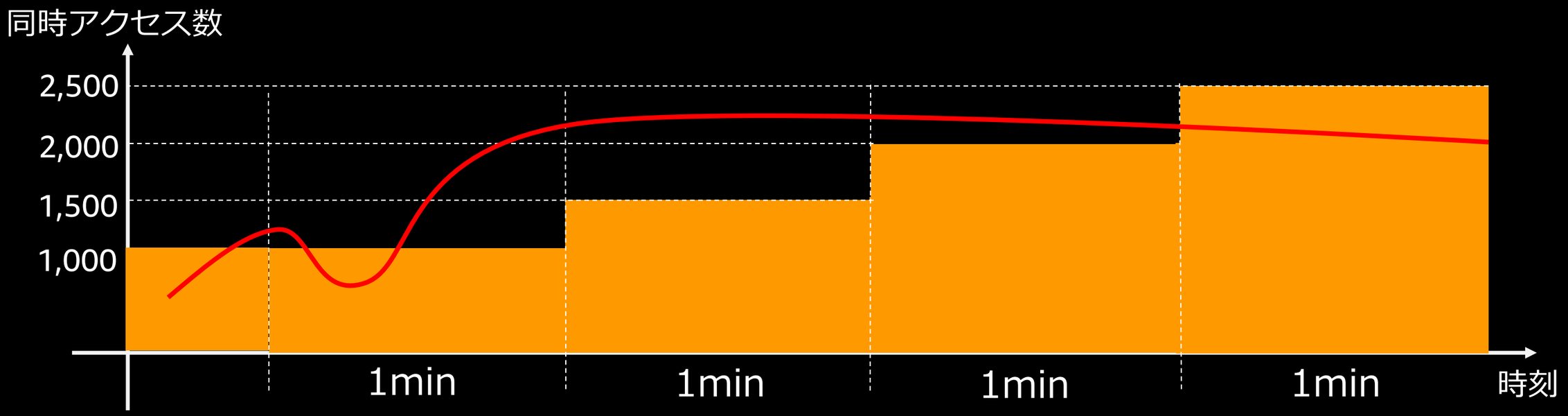
Lambdaの初期化（cold start） - スパイク時

Lambdaのスケーリング

同時実行数の上限はデフォルトで1,000（アカウント単位）

- 上限を緩和し同時実行数が1,000を超えた場合、上限まで500/分で追加

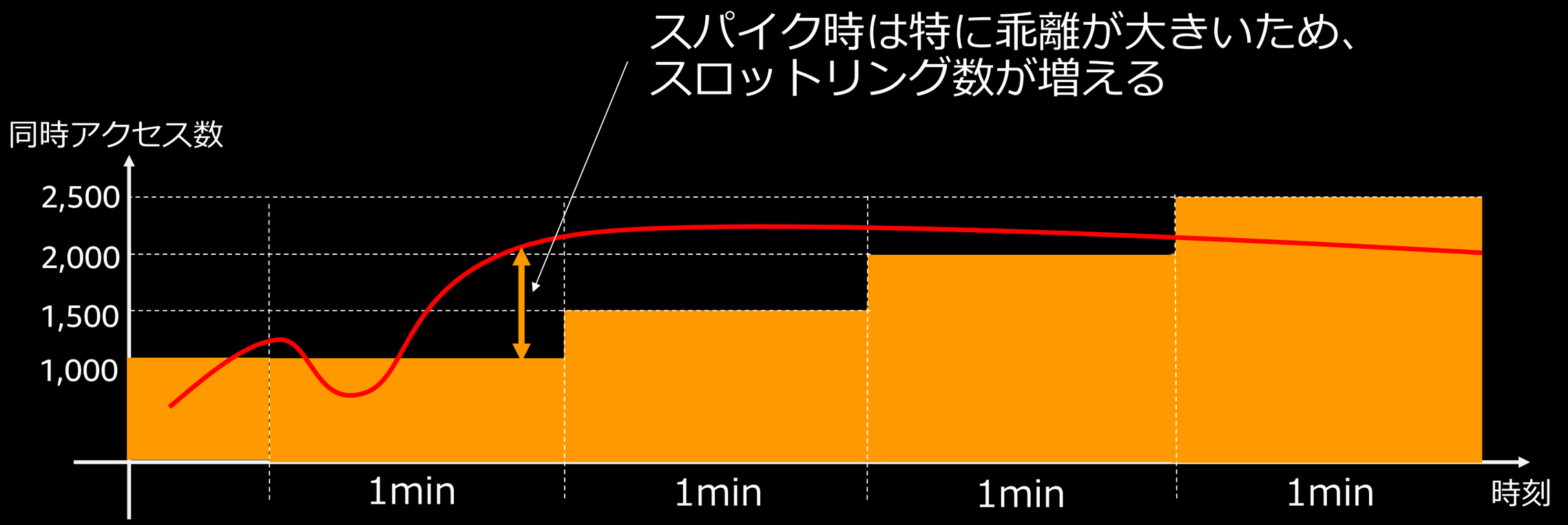
※東京リージョンの場合



※ 実際の挙動は厳密にこれに沿いません

https://docs.aws.amazon.com/ja_jp/lambda/latest/dg/scaling.html

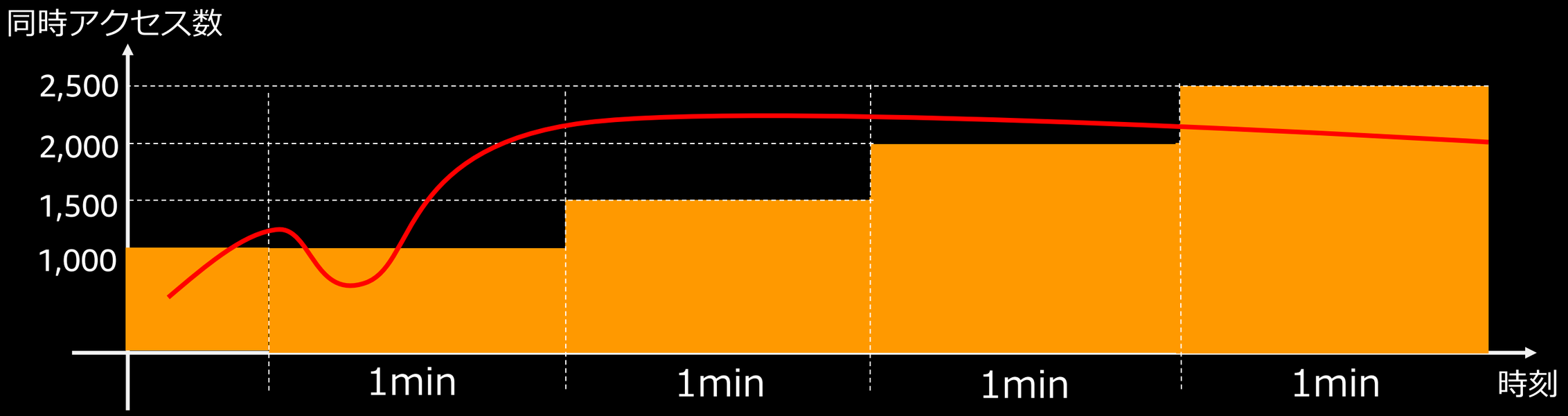
Lambdaのスケーリング



※ 実際の挙動は厳密にこれに沿いません

Lambdaのスケーリング

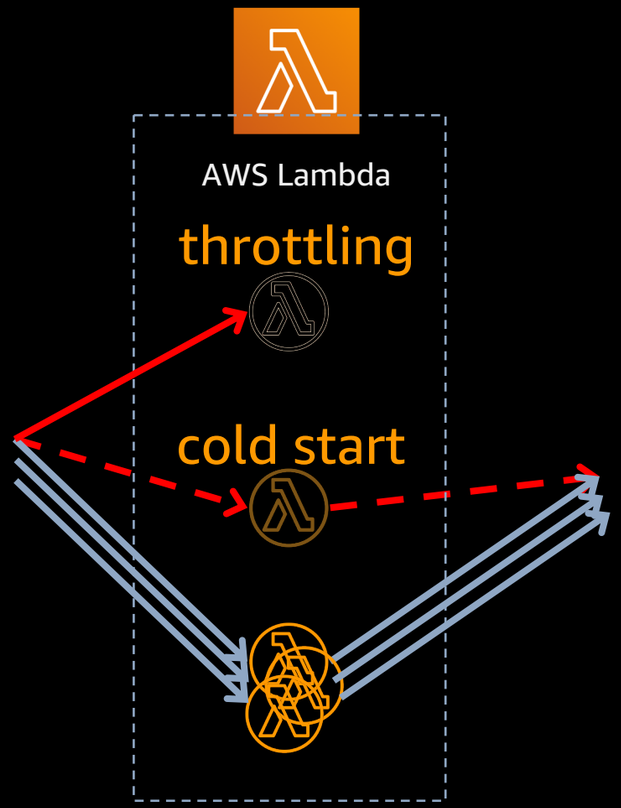
追加された関数インスタンスは初期化が必要 (cold start)



※ 実際の挙動は厳密にこれに沿いません

スパイク時のLambdaにおける注意点まとめ

- 同時実行数制限
- cold startによる実行時間の増加



大規模なアクセスや
スパイクに対応するには、
現在は設定が必要

スパイクに対処する

aws INNOVATE
ONLINE CONFERENCE

0. 事前準備

- クライアントでリトライ処理の実装
  - 問題は起きる前提で
- バージョン・エイリアスの作成
- 負荷テストを実施し、スロットリングやcold start発生の確認
  - スパイクとはいえアクセスの間には若干のラグがある
  - 負荷をかけるホストマシンは台数を増やす。単一だとDoSとみなされる、パフォーマンスが出ないなどの問題あり
- 必要な同時実行数の最大値、cold startの影響を見積もり
  - 余裕を持った見積もりを

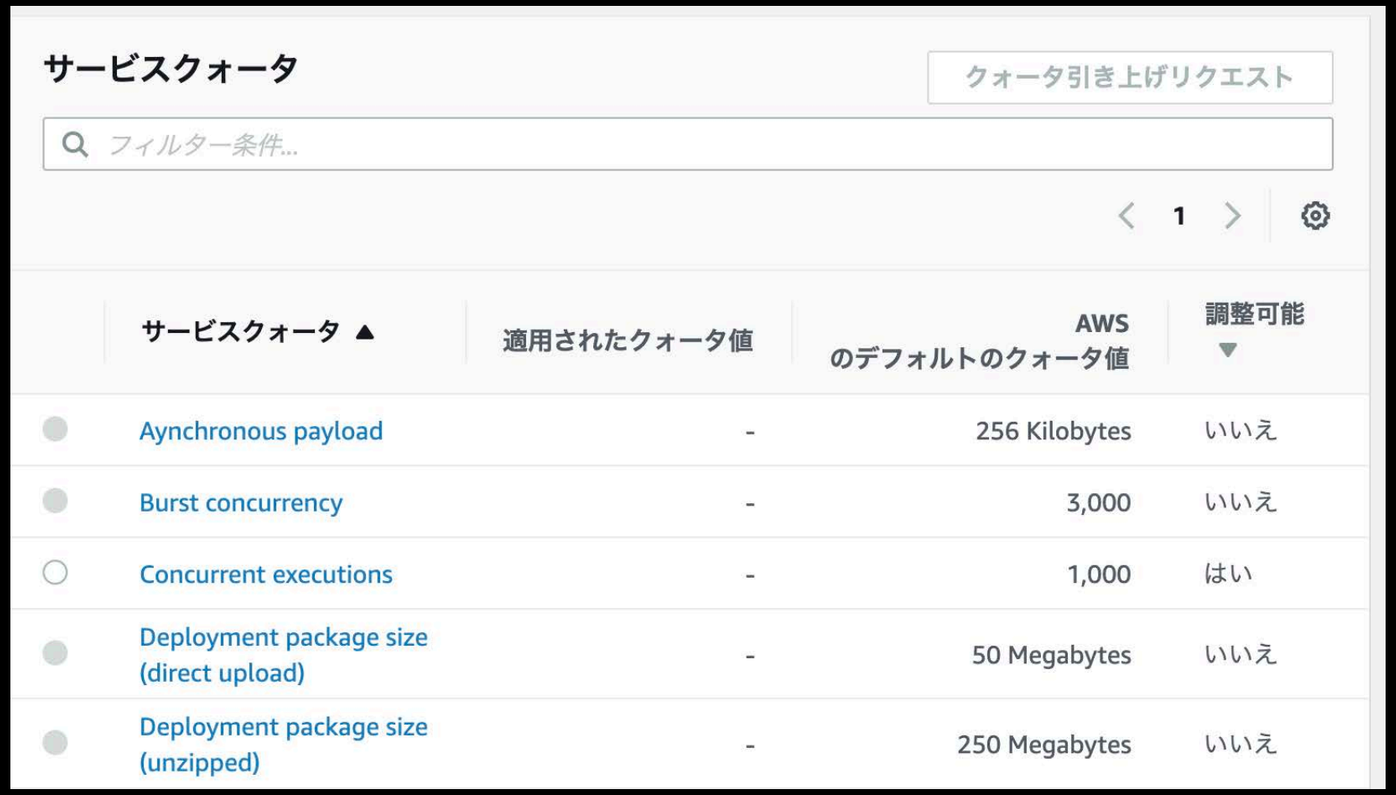
規模が大きくない場合、レイテンシ要件が厳しくない場合は対処不要

https://docs.aws.amazon.com/ja_jp/general/latest/gr/aws_service_limits.html

1. 同時実行数の上限を増やす

AWS Service Quotasで申請

- Lambdaの同時実行数が1,000を超える場合 (アカウント単位)



2. 同時実行数を増やす + 関数インスタンスを暖機

Provisioned Concurrencyを設定

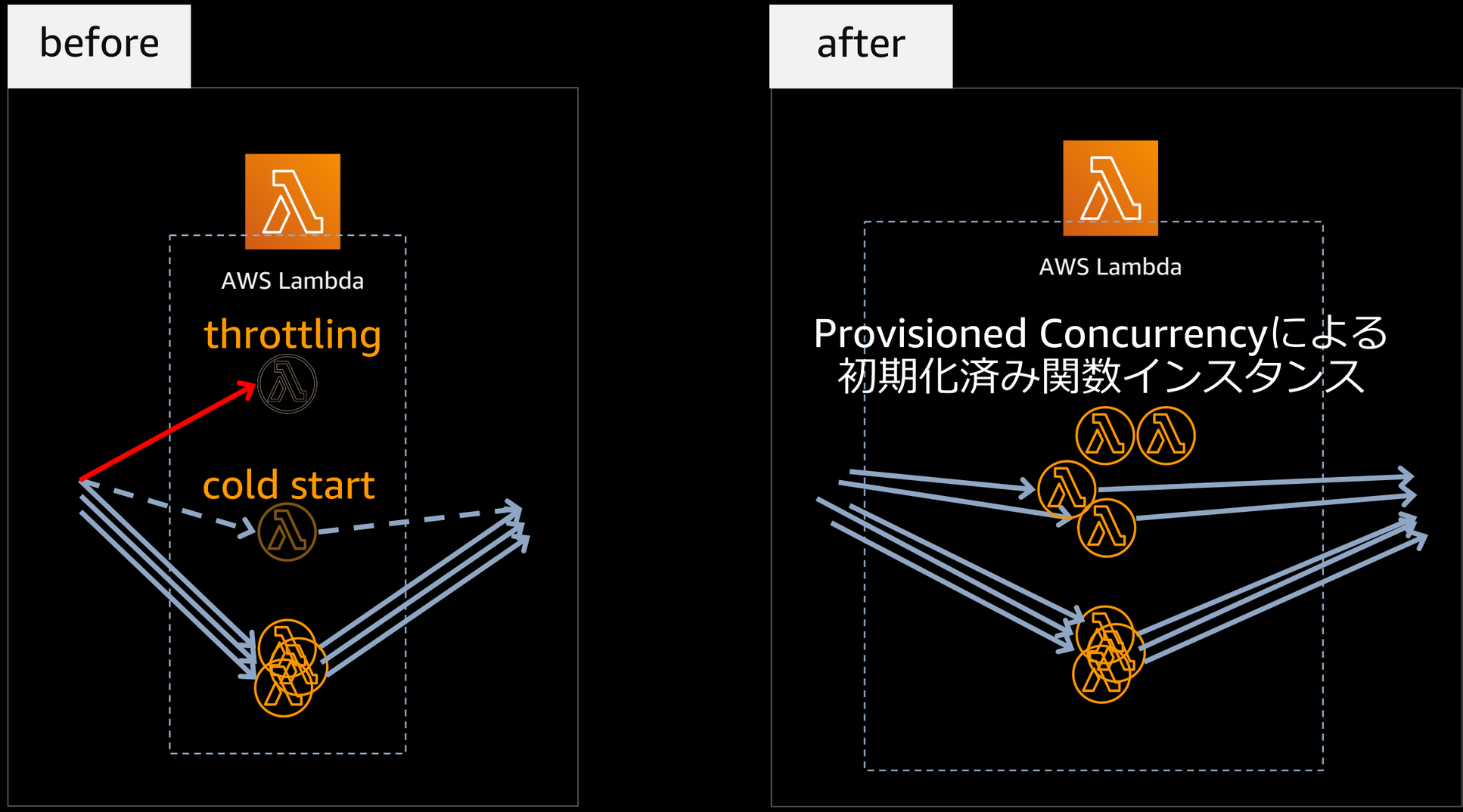
- インスタンスを初期化済みで用意
- プロビジョンした時間と数 + 実行についてお支払い

2. 同時実行数を増やす +関数インスタンスを暖機

設定可能な最大値 =
残同時実行上限-100

現在の利用可能数が足りない場合、手順1に戻る

Lambdaの活用イメージ (対策後のスパイク時)



Lambda以外の設定

Lambdaの周辺サービスも、設定が必要な場合あり

- API Gatewayで 10,000リクエスト/秒 を超える場合、上限緩和申請
- DynamoDBのキャパシティユニットを設定

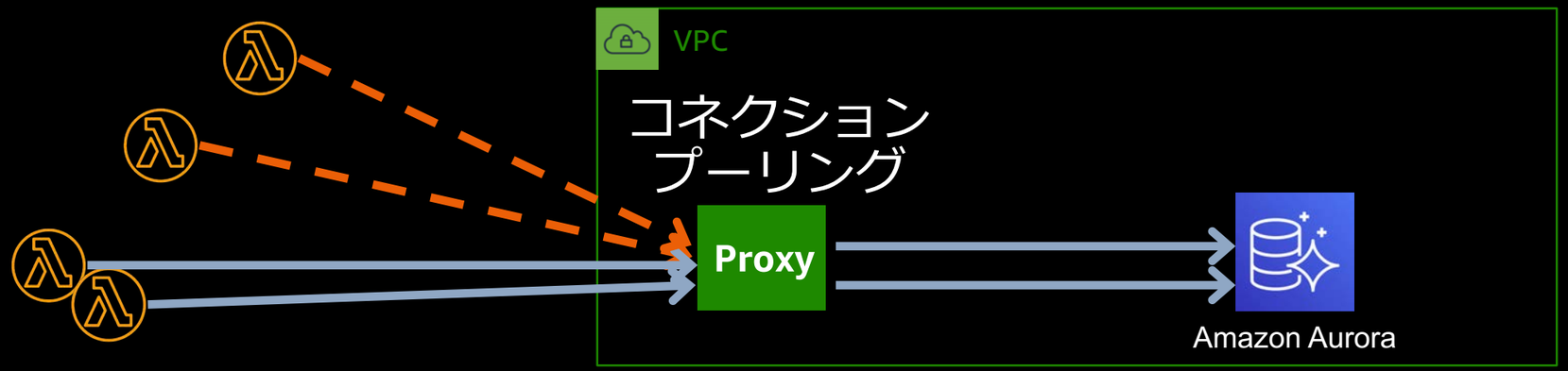
なるべく各種設定後にシステム全体の負荷テストを行う

詳細はAWS サービス別資料（AWS BlackBeltシリーズ）をご覧ください
https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/

RDBMSを使うパターンの注意点

1Lambda（または1関数インスタンス）ごとにコネクションを作成するとDBの負荷が高い

→ DBコネクションを使い回すためプロキシを挟む



コネクションプール数を超える（DBを使う）Lambdaの同時実行はできない

→ スロットリングや実行時間の増加・必要同時実行数の増加

Tips: 同時実行の予約 ≠ Provisioned Concurrency

同時実行の予約機能 = 他Lambdaの同時実行により、予期していた同時実行数が出ないことを防ぐ機能

関数インスタンスは準備されないため、**急激な同時実行に対処したりcold startを防ぐ機能ではない**

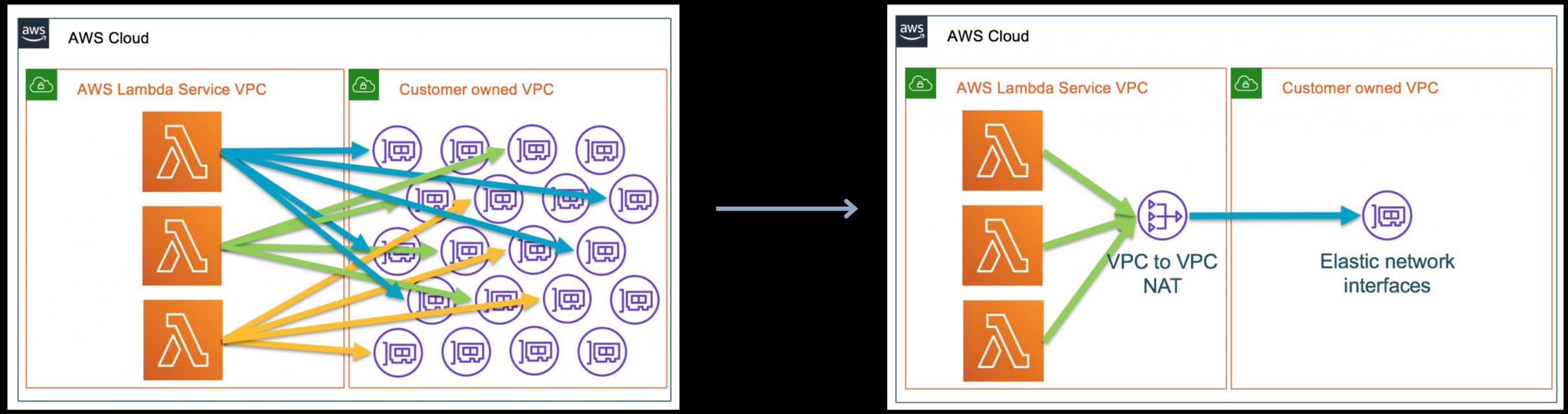
provisioned = 5
関数インスタンスを確保
さらに初期化済み

reserved = 5
同時実行数を確保
他のLambdaに使われない

Tips: VPCへの接続によるcold startは解消

2019年9月以前は、新規関数インスタンスがVPCに接続するごとにENIを作成

→ アップデートにより、**デプロイ時に共通のENIを作成**するようになった



https://aws.amazon.com/jp/blogs/news/announcing-improved-vpc-networking-for-aws-lambda-functions/

まとめ

aws INNOVATE
ONLINE CONFERENCE

まとめ

スパイクに対応するためには、サーバーレスサービスでも対策が必要

- 適切な予測とテスト、リトライ処理
- 上限の引き上げ（AWS Service Quotas）
- 同時実行数の確保（Provisioned Concurrency）
- Lambda周辺サービスの設定

以上を守って、
スパイクするワークロードでもサーバーレスの利点を活かそう！

クイズ

aws INNOVATE
ONLINE CONFERENCE

クイズ

ECサイトのAPIサーバーを、Lambdaを使用して作成しました。

デフォルト設定でLambdaをデプロイした後、以前取材されたテレビ番組でこのECサイトが取り上げられ、急激にアクセスが増加しました。結果、多数のスロットリングが発生し、機会損失が生まれてしまいました。

スロットリングを発生させないためには、どのような対処が必要だったでしょうか。

Thank you!

aws INNOVATE
ONLINE CONFERENCE