

公共システムに求められる耐障害性、 大量同時アクセスに対応する アーキテクチャ

原田 江海咲

パブリックセクター技術統括本部 アソシエイトソリューション アーキテクト
アマゾン ウェブ サービス ジャパン合同会社

自己紹介

原田 江海咲 (Emisa Harada)

- アソシエイト ソリューションアーキテクト :
 - 公共部門のお客様のクラウド活用を支援
- 好きなAWSサービス :



Amazon S3



Amazon Aurora



本セッションについて

対象者

- 公共機関（中央省庁、地方自治体、医療/ヘルスケア、教育事業者、非営利組織等）のIT部門のご担当者
- 公共機関におけるシステム開発に携わるシステムインテグレーターの方

ゴール

- AWS における耐障害性の考え方と、耐障害性を高めるアーキテクチャを理解する
- 大量同時アクセスに対処するアーキテクチャと、それを支えるサービスについて学ぶ

お話ししないこと

- 各サービスの機能の詳細
(AWS 公式ドキュメントや Black Belt オンラインセミナー資料をご確認ください)

本日のアジェンダ

- 公共システムならではの特性
- 耐障害性を高めるアーキテクチャ
- 大量同時アクセスを処理するアーキテクチャ
- まとめ
- ぜひやっていただきたいこと

公共システムならではの特性

- 災害時にこそ動かなければならない
 - 気象・防災情報を発信するサイト
 - 給付金システム
- 国民に向けたシステムはユーザーが膨大
 - 住民情報管理システム
 - 社会福祉関連システム
- 政策に基づいてアクセスが集中する
 - ワクチン接種予約サイト
 - 期限のある申請システム

これらの特性に対応したシステムとは？

- 災害等が発生してもサービス停止してはいけないシステム
- 短期間の大量アクセスに耐えうるシステム



高い耐障害性

大量同時アクセス
への対応

公共システムにおいてはこの 2 点に対応することが求められる

耐障害性を高めるアーキテクチャ

耐障害性の重要性

- 公共システムは **サービスを止めてはいけない**という大前提
- 昨今、**自然災害**のリスクは高まっている
 - 異常気象による台風や水害が頻繁に発生
 - 巨大地震のリスク
- クラウドも物理サーバー上で動いている以上、障害は発生しうる
 - **“Design for Failure”** (= 障害を見据えた設計) の考え方

オンプレミスで耐障害性を確保しようとする...

物理的に離れた場所での
サーバ調達や管理が大変

バックアップ体制を維持する
には莫大なコストがかかる



オンプレミスサーバー

オンプレミスで耐障害性を確保しようとする...

物理的に離れた場所での
サーバ調達や管理が大変



オンプレミスサーバー

AWS のグローバルインフラストラクチャ

世界中に **26** のリージョン

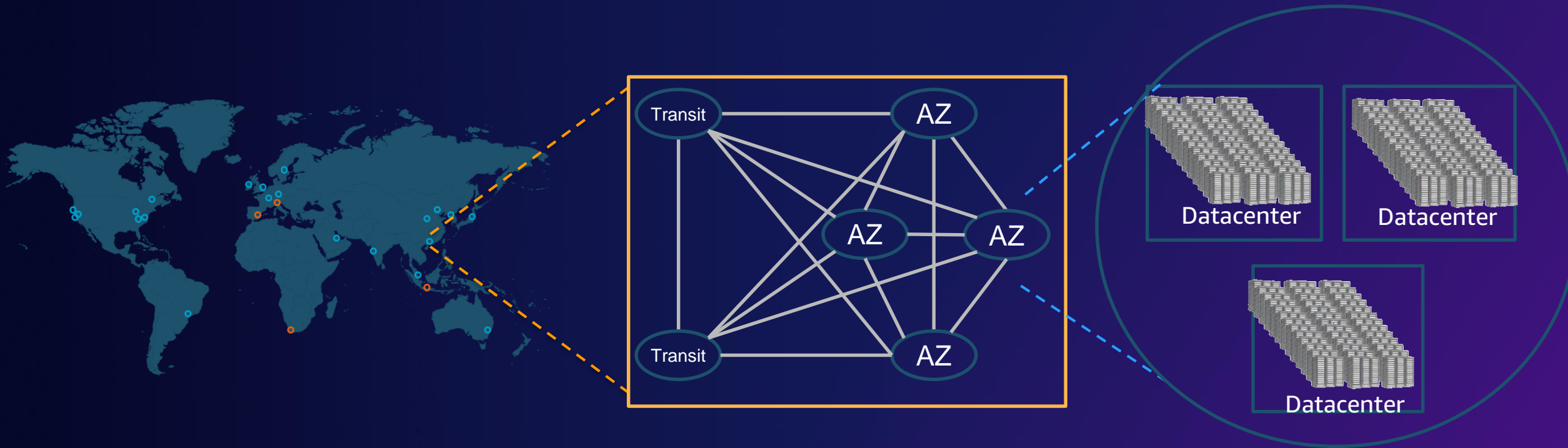
※2022 年 1 月時点

1. 米国東部 (バージニア北部)
2. 米国西部 (北カリフォルニア)
3. 米国西部 (オレゴン)
4. 欧州 (アイルランド)
5. アジアパシフィック(東京)
6. 南米 (サンパウロ)
7. アジアパシフィック (シンガポール)
8. アジアパシフィック (シドニー)
9. GovCloud (米国西部) *1
10. 中国 (北京)*2
11. 欧州 (フランクフルト)
12. アジアパシフィック (ソウル)
13. アジアパシフィック(ムンバイ)
14. 米国東部(オハイオ)
15. カナダ(中部)
16. 欧州(ロンドン)
17. 中国(寧夏) *2
18. 欧州(パリ)
19. アジアパシフィック (大阪)
20. GovCloud (米国東部) *1
21. 欧州(ストックホルム)
22. 香港特別自治区
23. 中東(バーレーン)
24. アフリカ(ケープタウン)
25. 欧州(ミラノ)
26. アジアパシフィック (ジャカルタ)



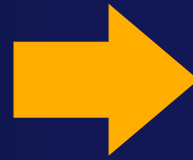
AWSのリージョンにおけるアベイラビリティゾーン (AZ)

- それぞれのリージョンは、複数の **アベイラビリティゾーン (AZ)** で構成されている
- AZ は自然災害やデータセンター単位の障害などによるリスクを最小化するように**地理的に影響を受けない十分離れた場所**にあり、独立した電源、空調、物理的なセキュリティを備えている



オンプレミスで耐障害性を確保しようとする...

物理的に離れた場所での
サーバ調達や管理が大変



AWS のグローバルインフラストラクチャ

- 世界中のリージョンをすぐに利用できる
- 物理的なサーバのメンテナンス不要



オンプレミスサーバー

オンプレミスで耐障害性を確保しようとする...


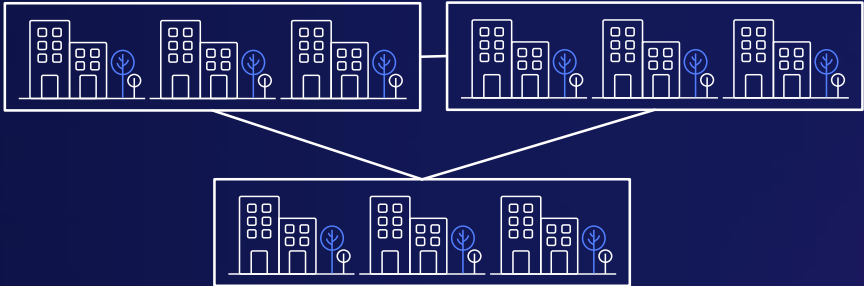

バックアップ体制を維持する
には莫大なコストがかかる



オンプレミスサーバー

AWS における冗長化の考え方

求められる**可用性とコスト**に見合った**アーキテクチャ**を選んで構築することが可能

シングル AZ	マルチ AZ	マルチリージョン
		
<ul style="list-style-type: none">• ディスクは冗長化され、単一障害に耐える• EC2 Auto Recovery、Auto Scalingなどによってサーバ障害にも対処可能	<ul style="list-style-type: none">• 複数AZ間で冗長化し、AZ 障害・中規模災害に耐える	<ul style="list-style-type: none">• 複数リージョン間で冗長化し、リージョン障害・大規模災害に耐える

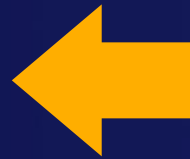
可用性 低
コスト 低



オンプレミスで耐障害性を確保しようとする...

可用性とコストに合わせて選べる冗長構成

- シングル AZ、マルチ AZ、マルチリージョン



バックアップ体制を維持するには莫大なコストがかかる



オンプレミスサーバー

耐障害性の高いアーキテクチャを構築するには？

RTO と RPO を定義することで、サービスの**耐障害性を測る指標**が明確化する



Recovery Point Objective (RPO)

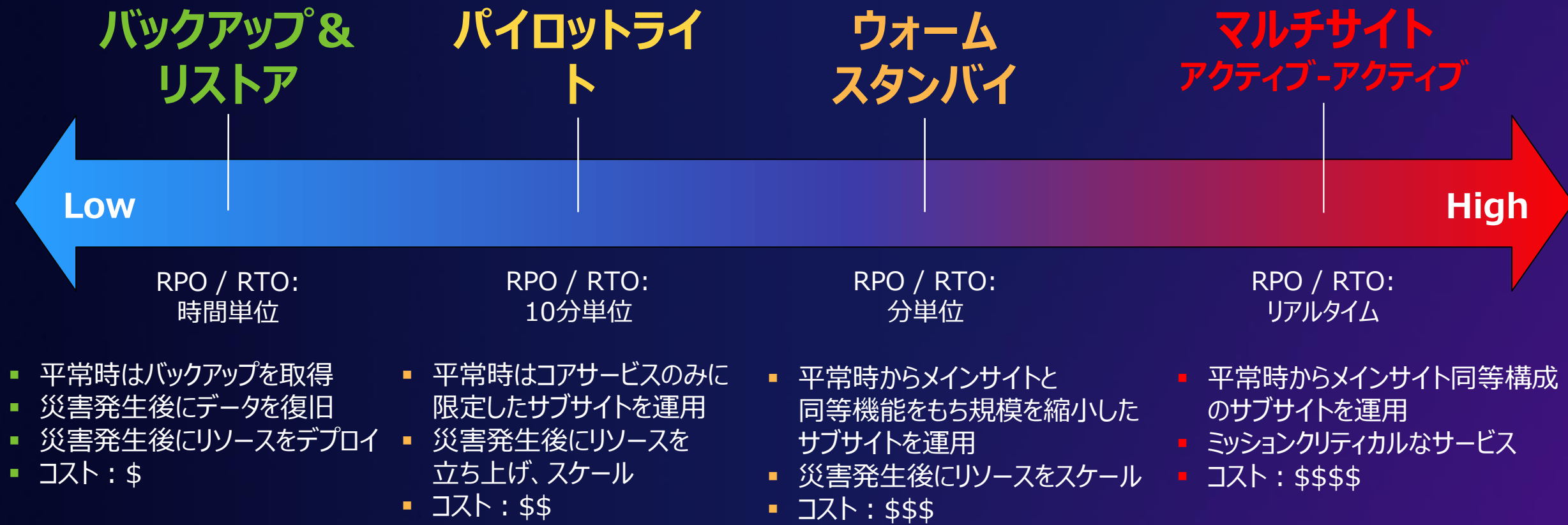
- 各データをどの時点まで戻す必要があるか？
- 例：障害発生直前時点まで、1 時間前まで、1 日前まで
- バックアップ・リストアの運用間隔やデータレプリケーションの技術選択に影響

Recovery Time Objective (RTO)

- システムの復旧にどれくらい時間がかかるか？
- 例：1 分以内、1 時間以内、1 週間以内
- データリストアや、システム再起動等の技術選択に影響

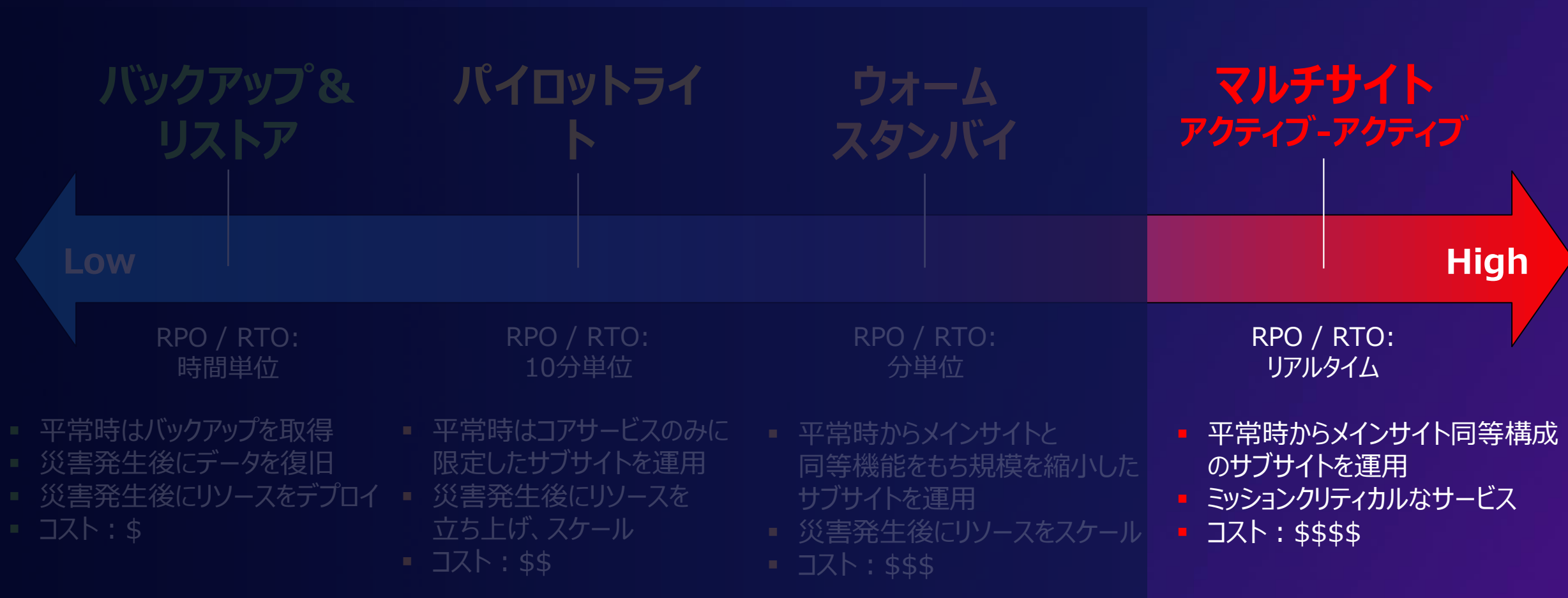
RTO / RPO に応じた 4 つの復旧シナリオ

RTO と RPO に応じた復旧シナリオを検討する



RTO / RPO に応じた 4 つの復旧シナリオ

RTO と RPO に応じた復旧シナリオを検討する

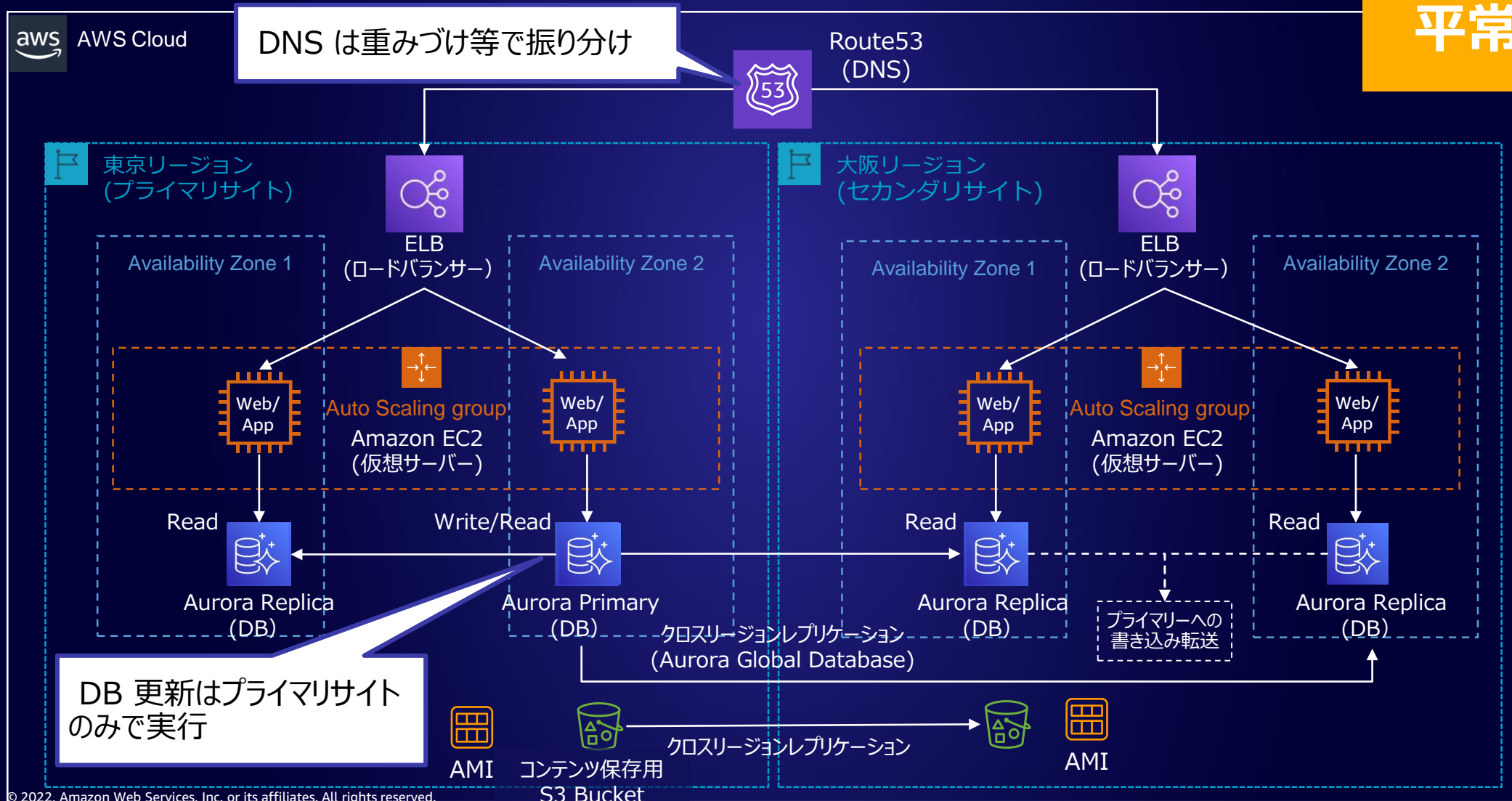


マルチサイト アクティブ-アクティブ

可用性 **99.999%**、復旧時間 **1 分未満**

フェイルオーバーの準備が整っている**本番サイトの完全な複製**。ミッションクリティカルなシステムに有効。

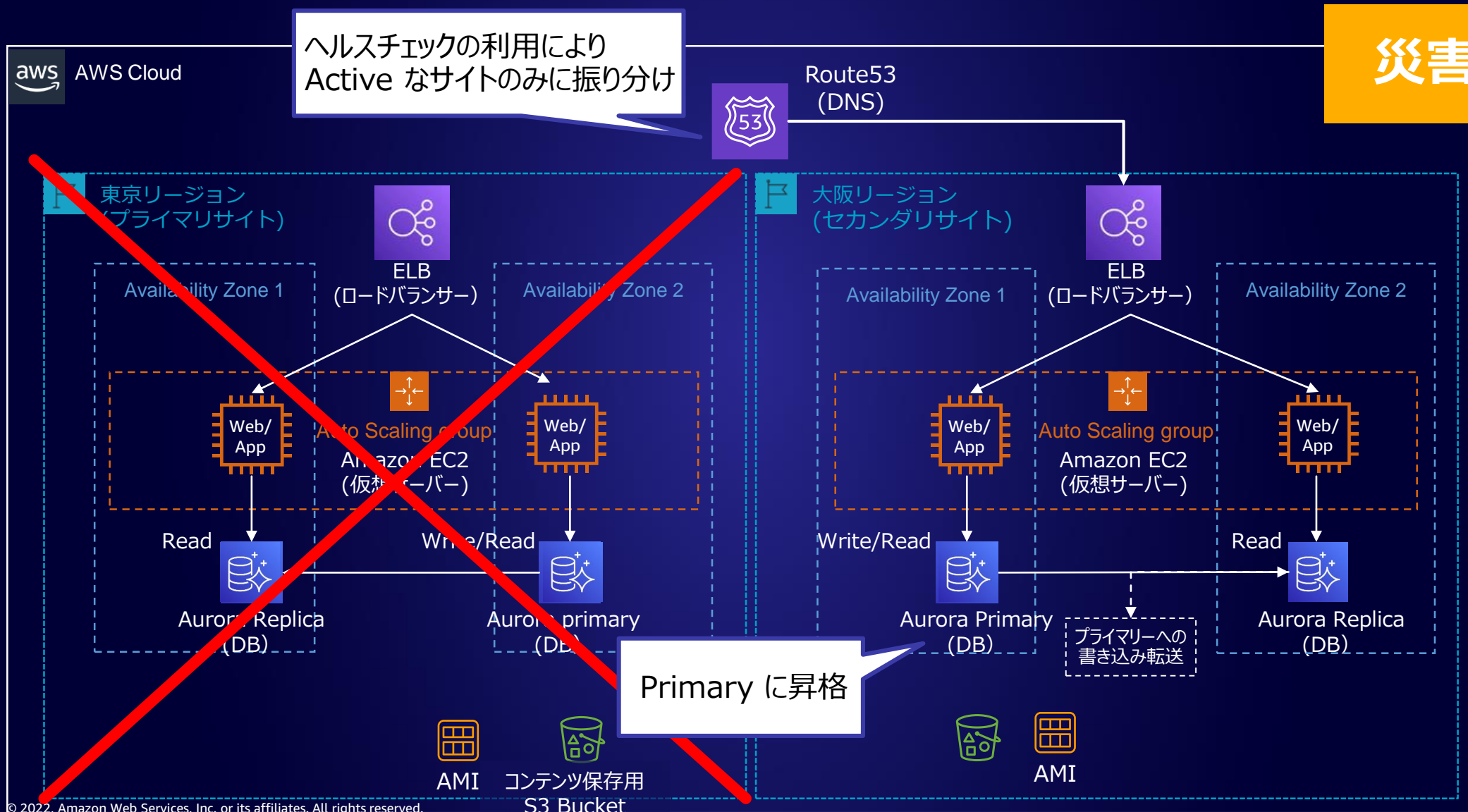
平常時



マルチサイト アクティブ-アクティブ

可用性 **99.999%**、復旧時間 **1 分未満**

災害時



耐障害性の高いシステムを構築するには？

- RTO / RPOを定義する
 - 耐障害性を測る指標の明確化
- RTO / RPOを考慮したアーキテクチャの選定
 - バックアップ&リストア
 - パイロットライト
 - ウォームスタンバイ
 - マルチサイト アクティブ-アクティブ

現実のアプリケーションにおいて、
RTO や RPO を満たせているか測定・管理するには？

AWS Resilience Hub

2021 / 11

NEW

アプリケーションの RTO と RPO を測定し、耐障害性（**レジリエンス**）の定義、追跡、管理を支援するサービス

- **AWS Well-Architected Framework** に沿って潜在的な耐障害性の弱点を明らかにする
- RPO / RTO を満たすための構成・運用における **推奨事項を提案**
- **Amazon CloudWatch** や **AWS Fault Injection Simulator (FIS)** と連携し、耐障害性に関するアラートやインサイトを継続的に可視化

マルチAZ構成のアプリケーションの RPO / RTO を測定した例

Assessment-report-dmc5ymoi454  Policy breached


▼ Overview

Application WebServer
Assessed on November 8, 2021, 20:33 (UTC-08:00)

Results

アプリケーション、インフラ、AZ、リージョン単位での災害それぞれに対する評価結果が表示される

RTO		
Disruption type	Targeted	Estimated
Application	2h	 unrecoverable
Infrastructure	5m	 2m
Availability Zone	5m	 2m
Region	-	-

RPO		
Disruption type	Targeted	Estimated
Application	15m	 unrecoverable
Infrastructure	5m	 0s
Availability Zone	5m	 0s
Region	-	-

Customer Application RTO and RPO [Info](#)







▼ Application 

RTO Targeted 2h

Estimated unrecoverable

▼ Breaches

Estimated RTO and RPO

Component	Targeted RTO	Estimated RTO	Targeted RPO	Estimated RPO
computeappcomponent-ec2	2h	 15m	15m	 0s
databaseappcomponent-rds	2h	 unrecoverable	15m	 unrecoverable
networkingappcomponent-natgw	2h	 0s	15m	 0s
storageappcomponent-s3bucket	2h	 unrecoverable	15m	 unrecoverable

RTO / RPOを満たせていないリソースを特定



前半のまとめ：耐障害性を高めるアーキテクチャ

- AWS のグローバルインフラストラクチャ
 - 世界中に 26 のリージョン
- 可用性とコストに合わせて選べる冗長化パターン
 - シングル AZ, マルチ AZ, マルチリージョン
- RTO と RPO を考慮したアーキテクチャ
 - マルチサイト アクティブ-アクティブ構成によりミッションクリティカルなシステムにも対応可能
 - AWS Resilience Hub の活用

大量同時アクセスを処理するアーキテクチャ

大量同時アクセスに対応する重要性

- 公共システムでは 一定期間に集中アクセスが発生するケースが多い
 - ワクチン接種予約サイト
 - 災害時の給付金・寄付金システム
 - 教育現場で利用されるデジタル教科書
- 自然災害や障害に備えて耐障害性を高めたとしても、大量同時アクセスによる負荷に耐えきれず サービス提供が止まってしまう可能性がある

大量同時アクセスを処理する上での 2 つの考え方



1. 負荷に応じてリソースを
スケーリングさせる



2. バックエンドで受ける
トラフィック量を制御

大量同時アクセスを処理する上での 2 つの考え方



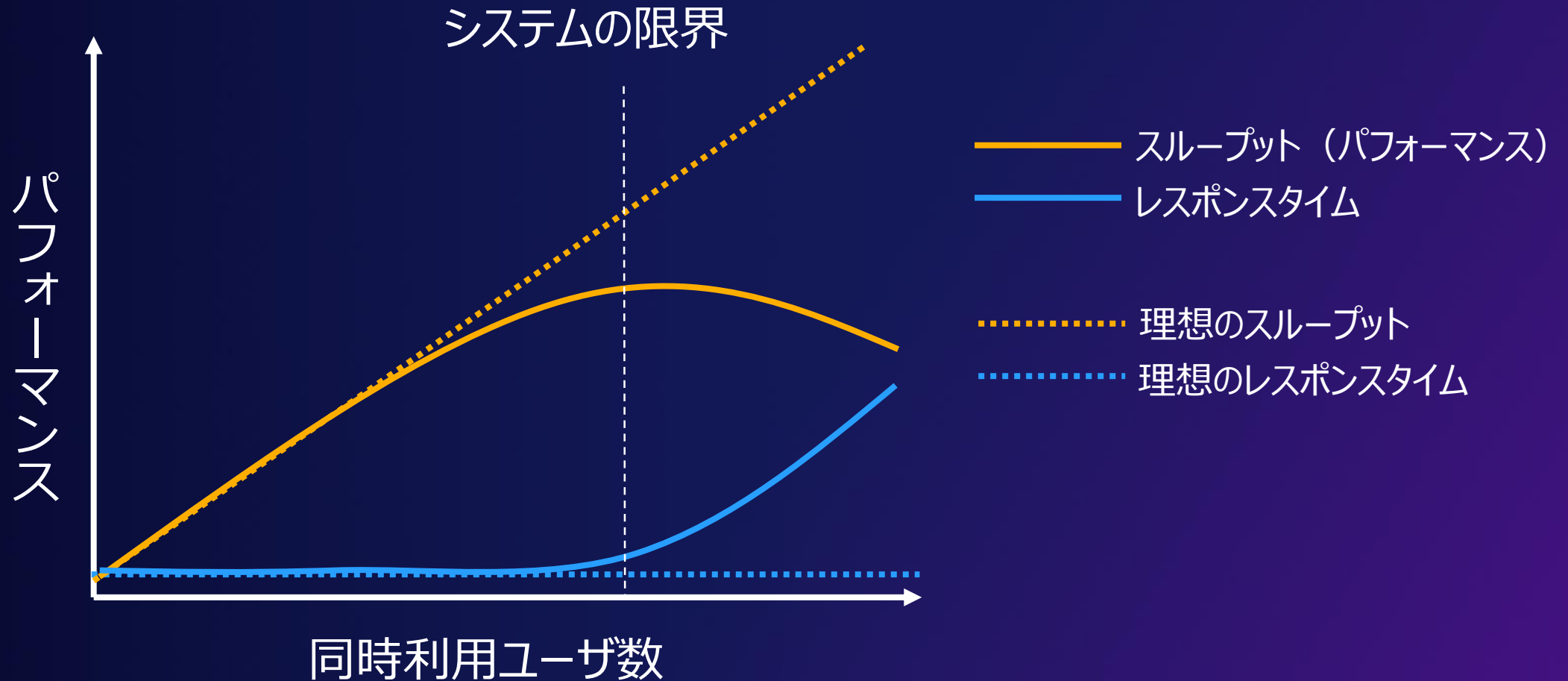
1. 負荷に応じてリソースを
スケーリングさせる



2. バックエンドで受ける
トラフィック量を制御

1. 負荷に応じてスケールするシステムとは？

利用負荷 や 利用規模（ユーザー数）が増えることに対応できるシステム



大量同時アクセスを処理する上での 2 つの考え方



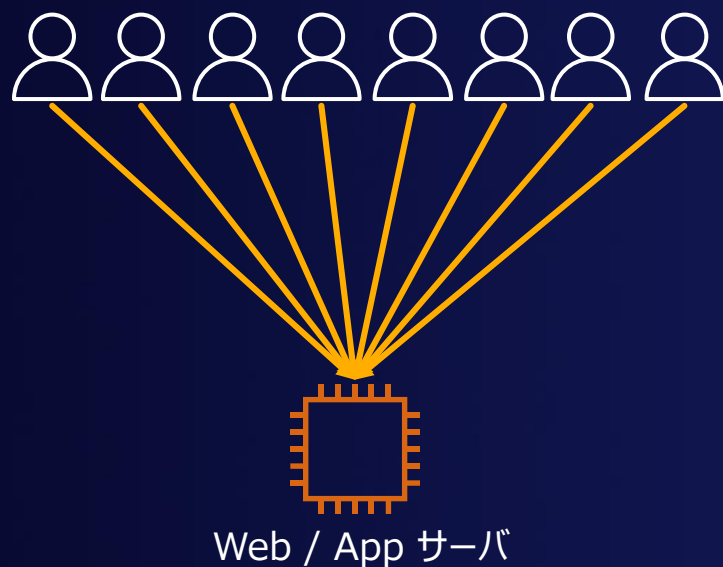
1. 負荷に応じてリソースを
スケーリングさせる



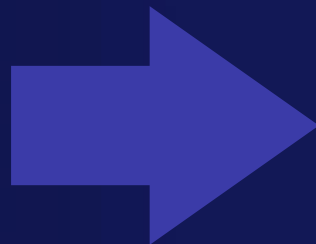
2. バックエンドで受ける
トラフィック量を制御

2. トラフィック量の制御とは？

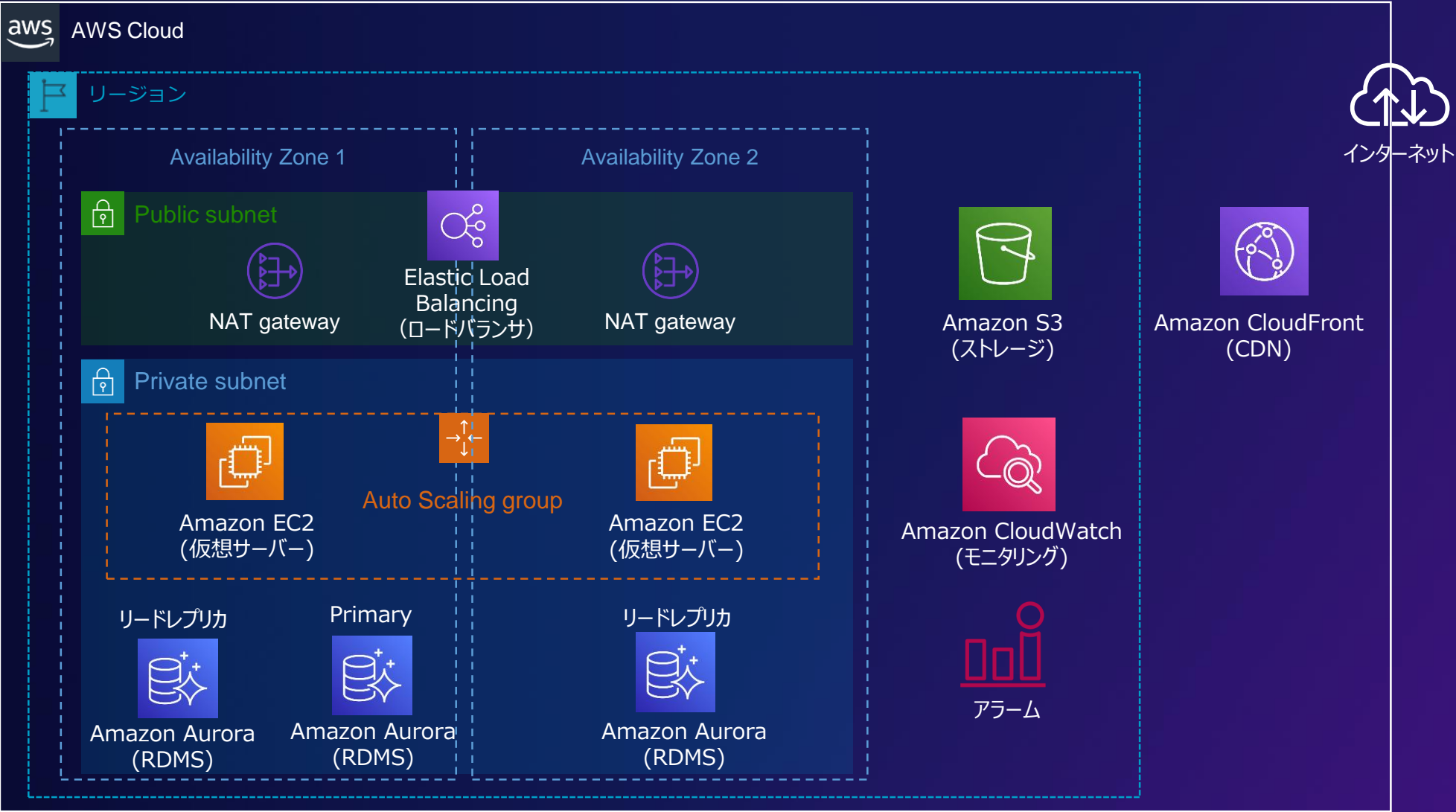
大量のリクエストをサーバーの前段となるサービスで一時的に吸収すること



リクエストに比例して負荷が増大し、
スパイクに対応しきれなくなる



大量同時アクセスを処理するアーキテクチャ例



大量同時アクセスを処理する上での 2 つの考え方

1. 負荷に応じてリソースをスケーリングする対策

- a. 静的コンテンツの処理をオフロード
- b. 仮想サーバを動的にスケーリング
- c. データベースの読み込み処理をスケーリング

2. トラフィック量を制御する対策

- a. エッジロケーションでのフィルタリング
- b. キューイングを用いたバッファリング

1-a. 静的コンテンツの処理をオフロード

- 大量のアクセスを処理するためには、**不要なトラフィックをバックエンドに到達させない**仕組みが必要
 - Webコンテンツには、**変化しない静的なデータ**が多く含まれる
(画像、動画、HTML / CSS、JavaScript 等のファイル)
 - 同じデータを何度も取得するのはサーバーリソースの無駄な消費



- 静的コンテンツを Web サーバーから切り離す
→ **Amazon S3**
- 一度取得したコンテンツをキャッシュする
→ **Amazon CloudFront**

1-a. 静的コンテンツの処理をオフロード



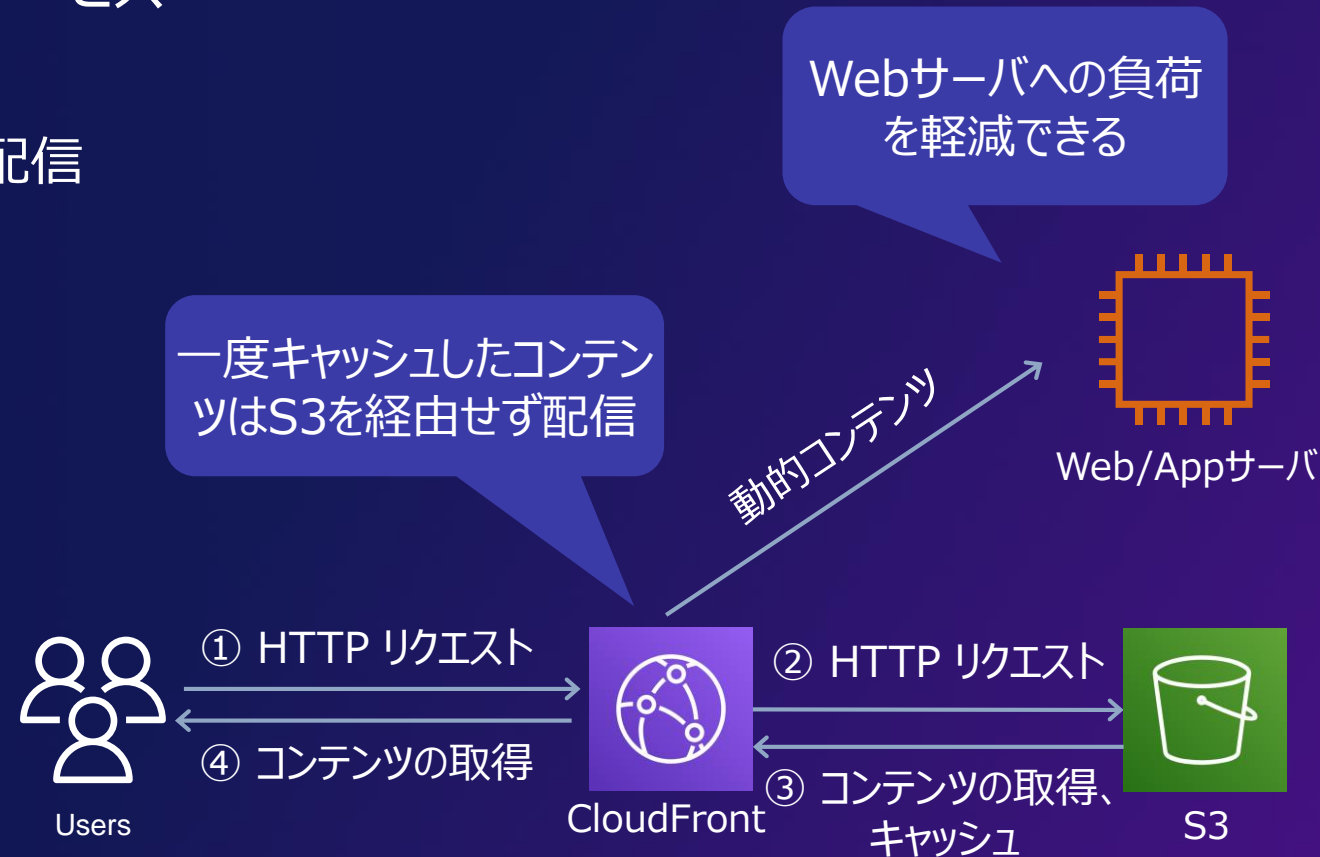
Amazon CloudFront

- CDN (Contents Delivery Network) サービス
- サイトの**高速化**、サーバの**負荷軽減**
 - 世界中の **エッジロケーション** からコンテンツを高速配信



Amazon S3

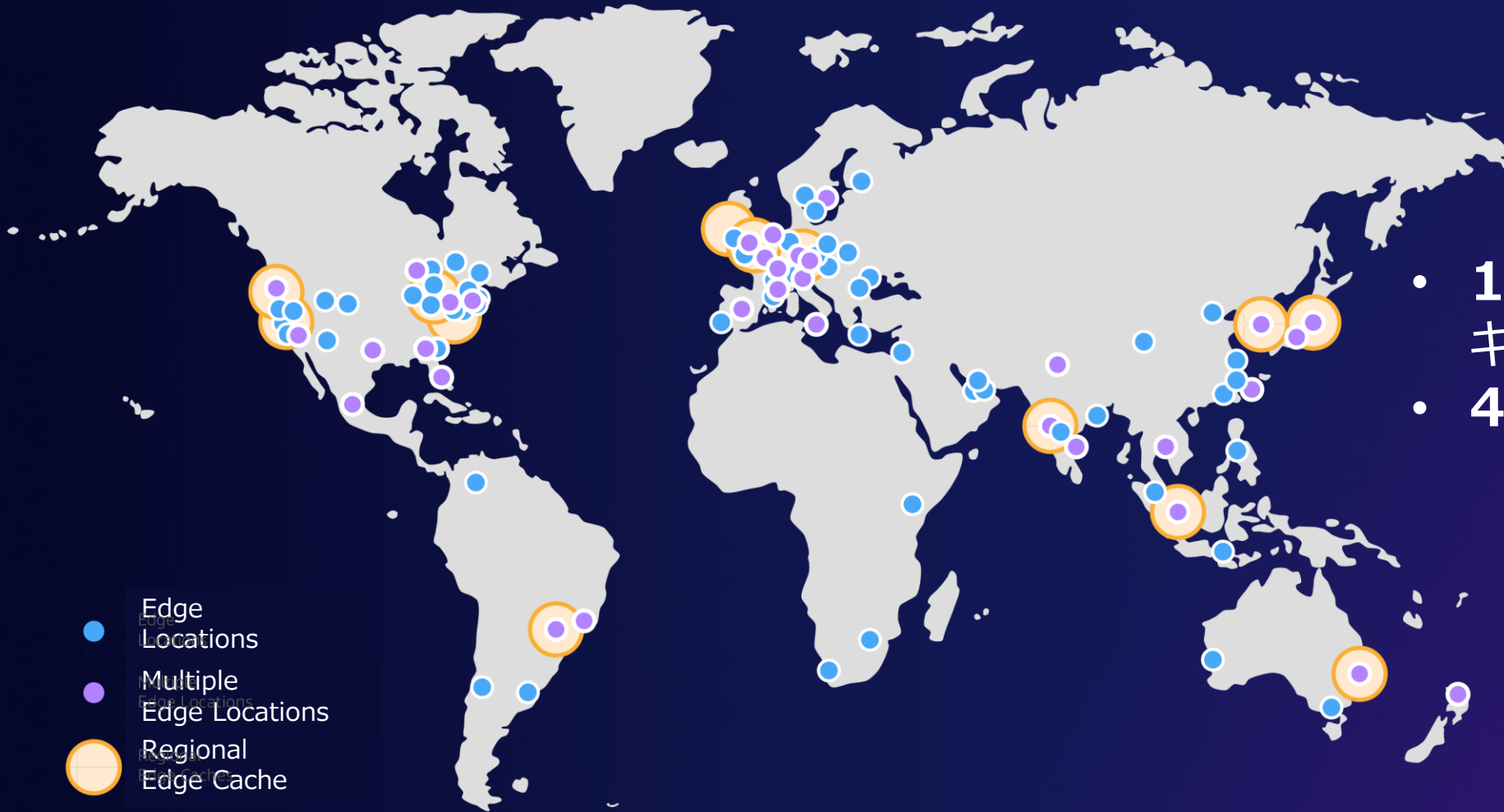
- スケーラブルで耐久性のあるストレージ
 - 99.999999999% の**高い耐久性**
 - データ容量に依存しない性能
- 静的コンテンツをホストできる
→ **CloudFront** を使うことで**高速化**できる



エッジロケーション

※2022年1月時点

310+ の POP (Point Of Presence)



- **13** のリージョン別エッジキャッシュ
- **47** か国 **90** 都市に展開

大量同時アクセスを処理する上での 2 つの考え方

1. 負荷に応じてリソースをスケーリングする対策

- a. 静的コンテンツの処理をオフロード
- b. 仮想サーバを動的にスケーリング
- c. データベースの読み込み処理をスケーリング

2. トラフィック量を制御する対策

- a. エッジロケーションでのフィルタリング
- b. キューイングを用いたバッファリング

1-b. 仮想サーバーを動的にスケーリング

- ユーザー数やアクセス数に応じたリソースを用意する必要がある
- 大量同時アクセスが発生しうるシステムでは、**事前に予測を立てるのが難しい**
 - 過剰にサーバーを用意したことによる**不要なコストの発生**
 - リソース不足により、必要なときに**サービスを提供できない**



- 実際の需要をモニタリング
 - **Amazon CloudWatch**
- 需要に合わせてリソースを増減させる
 - **AWS Auto Scaling**

1-b. 仮想サーバーを動的にスケーリング



Amazon CloudWatch

- AWS内のリソース や アプリケーションの **モニタリングサービス**
 - メトリクス や ログを収集することで、**リソース使用率やパフォーマンスを可視化**

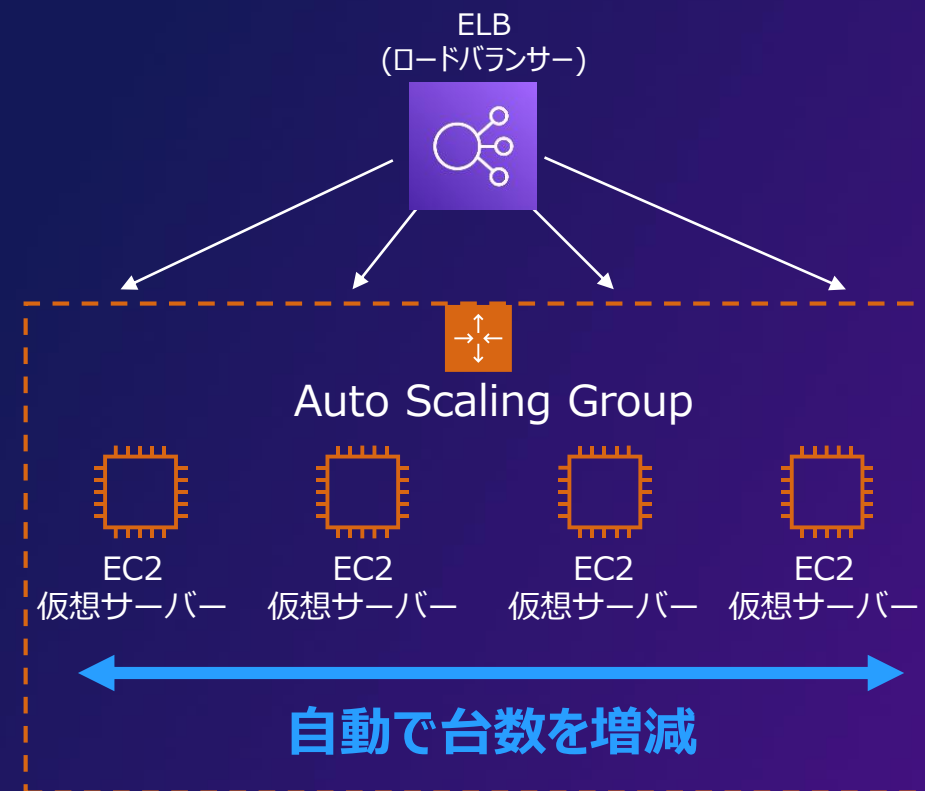


AWS Auto Scaling

- 定義した**ポリシー**に従って**仮想サーバーの台数を増減**

ポリシーの例：

- CPU 使用率 80% 以上が 5 分間続いたらサーバを 2 台増やす
- 毎朝 09:00 AM にサーバを 3 台増やす



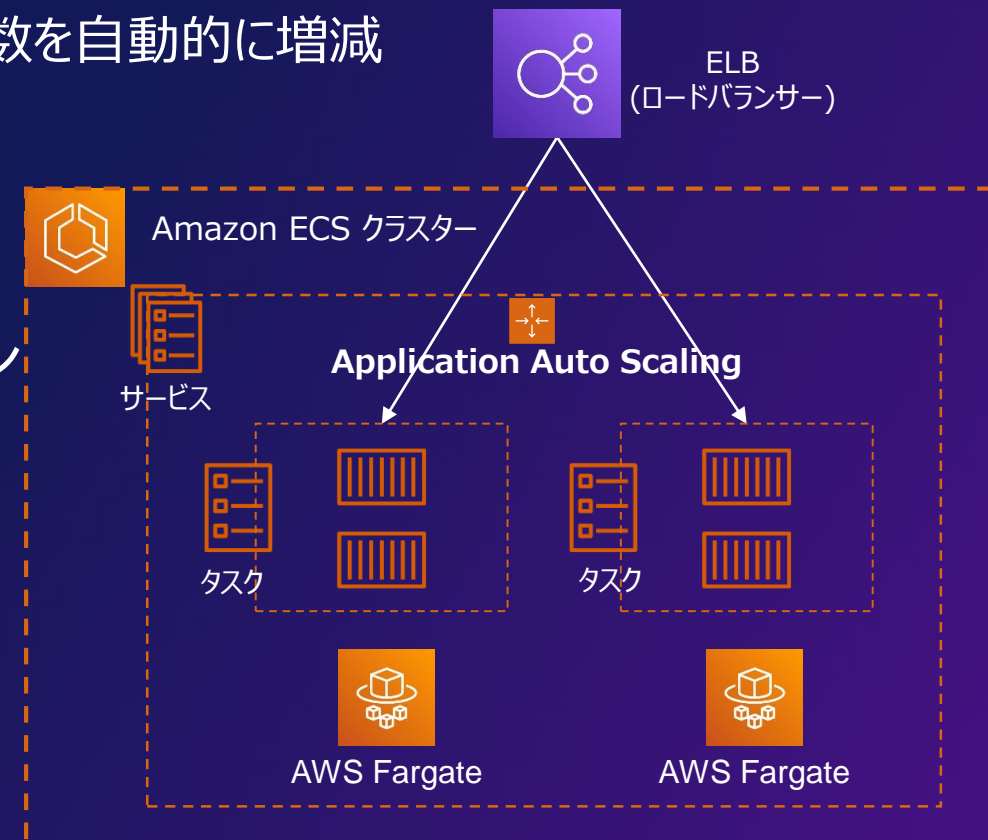
1-b. (参考) コンテナによるスケーリング

Amazon Elastic Container Service (ECS)

- クラウドでコンテナを本番環境利用するためのオーケストレーター
 - 多様なワークロードをサポートする「タスク」「サービス」というシンプルな構成要素
 - Application Auto Scaling** を活用してサービスの必要タスク数を自動的に増減

AWS Fargate

- サーバー管理不要のコンテナ実行コンピューティングエンジン
 - EC2 インスタンスの管理不要
 - AWS 側で**スケーリングを自動管理**



大量同時アクセスを処理する上での 2 つの考え方

1. 負荷に応じてリソースをスケーリングする対策

- a. 静的コンテンツの処理をオフロード
- b. 仮想サーバを動的にスケーリング
- c. データベースの読み込み処理をスケーリング

2. トラフィック量を制御する対策

- a. エッジロケーションでのフィルタリング
- b. キューイングを用いたバッファリング

1-c. データベースの読み込み処理をスケーリング

- サーバーの増加に伴う**データベースへのリクエスト増加対策**が必要
- 参照系クエリによるデータベースの負荷を削減することが重要
 - 多くの Web アプリケーションにおいては、更新系クエリよりも**参照系クエリ**の方が多い



- リードレプリカへの参照系クエリのオフロード
→ **Amazon Aurora リードレプリカ**

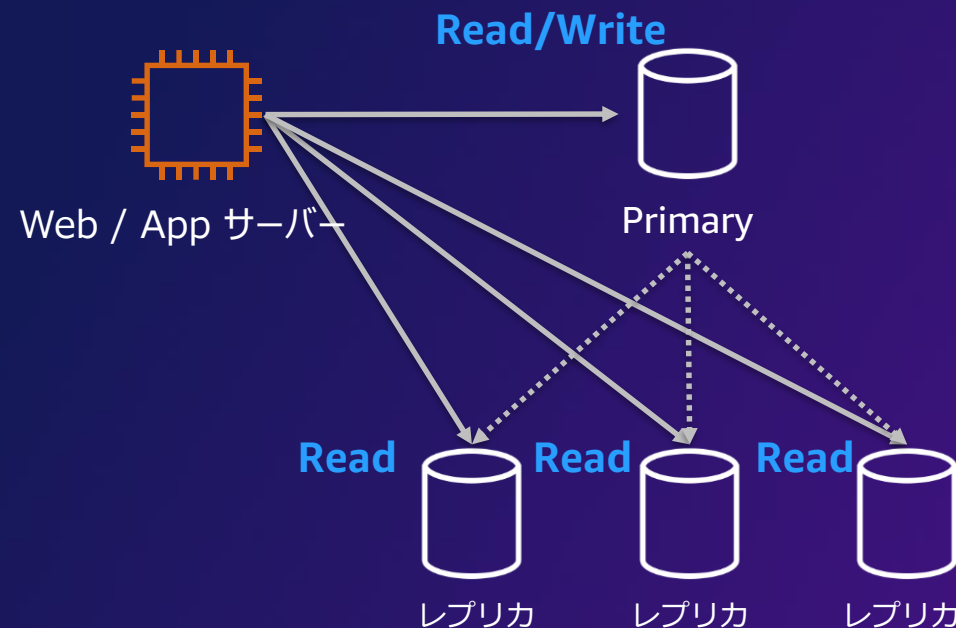
1-c. データベースの読み込み処理をスケーリング

Amazon Aurora

- MySQL, PostgreSQL と互換性のある**完全マネージド型 RDBMS**
- **コマーシャルデータベースの性能と可用性**を 1/10 のコストで実現
 - 3 つのAZにわたり、6 個のコピーを保持

Aurora レプリカ

- 3つのAZで**最大15個**の昇格可能なリードレプリカ
- 20～40ミリ秒のレプリカラグ
- **Auto Scaling** による自動増減が可能
 - 平均CPU使用率・平均接続数に応じて**Reader** を自動増減



大量同時アクセスを処理する上での 2 つの考え方

1. 負荷に応じてリソースをスケーリングする対策

- a. 静的コンテンツの処理をオフロード
- b. 仮想サーバを動的にスケーリング
- c. データベースの読み込み処理をスケーリング

2. トラフィック量を制御する対策

- a. エッジロケーションでのフィルタリング
- b. キューを用いたバッファリング

2-a. エッジロケーションでのフィルタリング

- スケーリングをしても、**急激なスパイク**には耐えきれない
 - 仮想サーバが起動するまでに数分かかる
- 仮想サーバの前のレイヤーで**トラフィックを吸収**し、大量アクセスからサーバーを保護する必要がある



- エッジロケーションでトラフィックをフィルタリングする
→ **CloudFront Functions**

2-a. エッジロケーションでのフィルタリング

2021 / 5

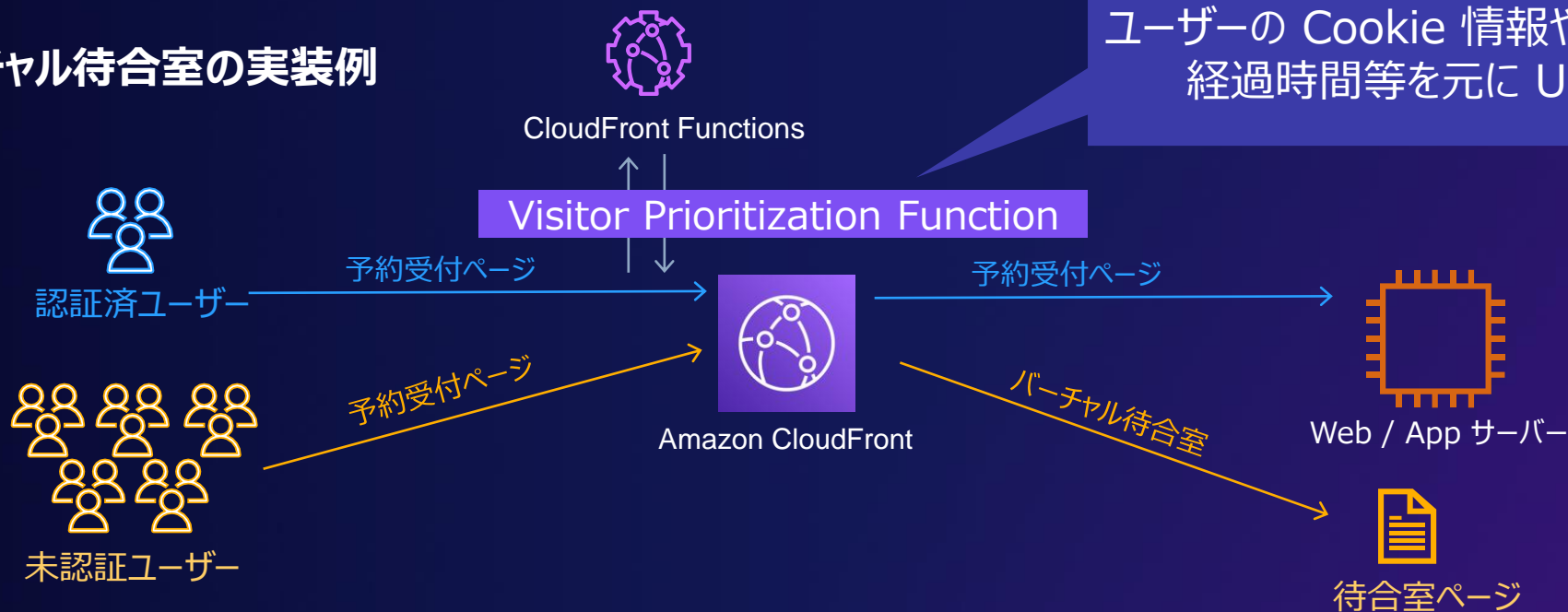
NEW



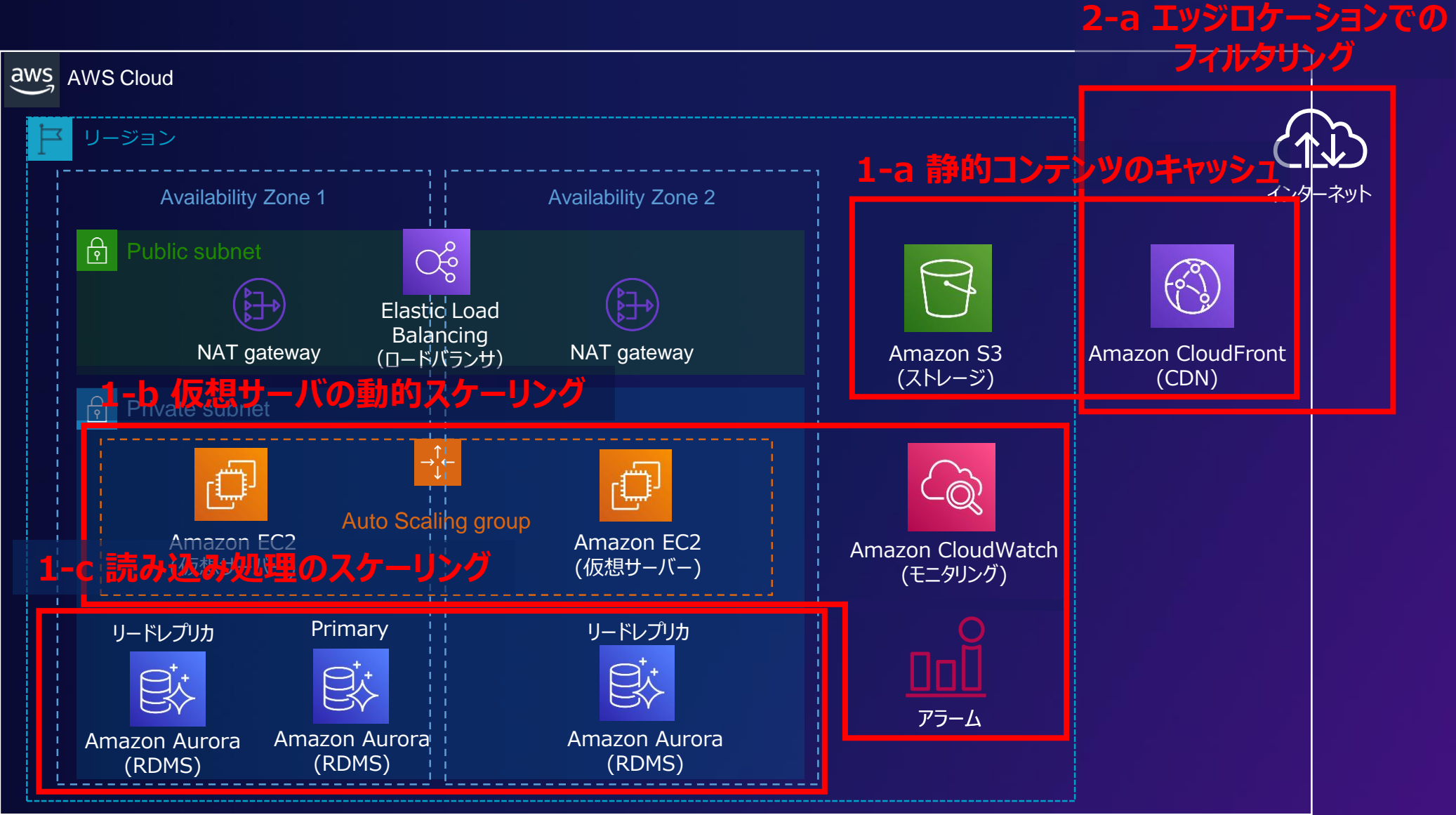
CloudFront Functions

- 軽量の JavaScript コードを **CloudFront エッジロケーション** で実行
 - **予測不可能でスパイクの多い**トラフィックをサポート (数百万リクエスト/秒まで拡張)
 - Lambda@Edge の 1/6 のコスト、より低いレイテンシー

バーチャル待合室の実装例



大量同時アクセスを処理するアーキテクチャ例



事例紹介 – 両備システムズ様

新型コロナワクチン接種予約システム

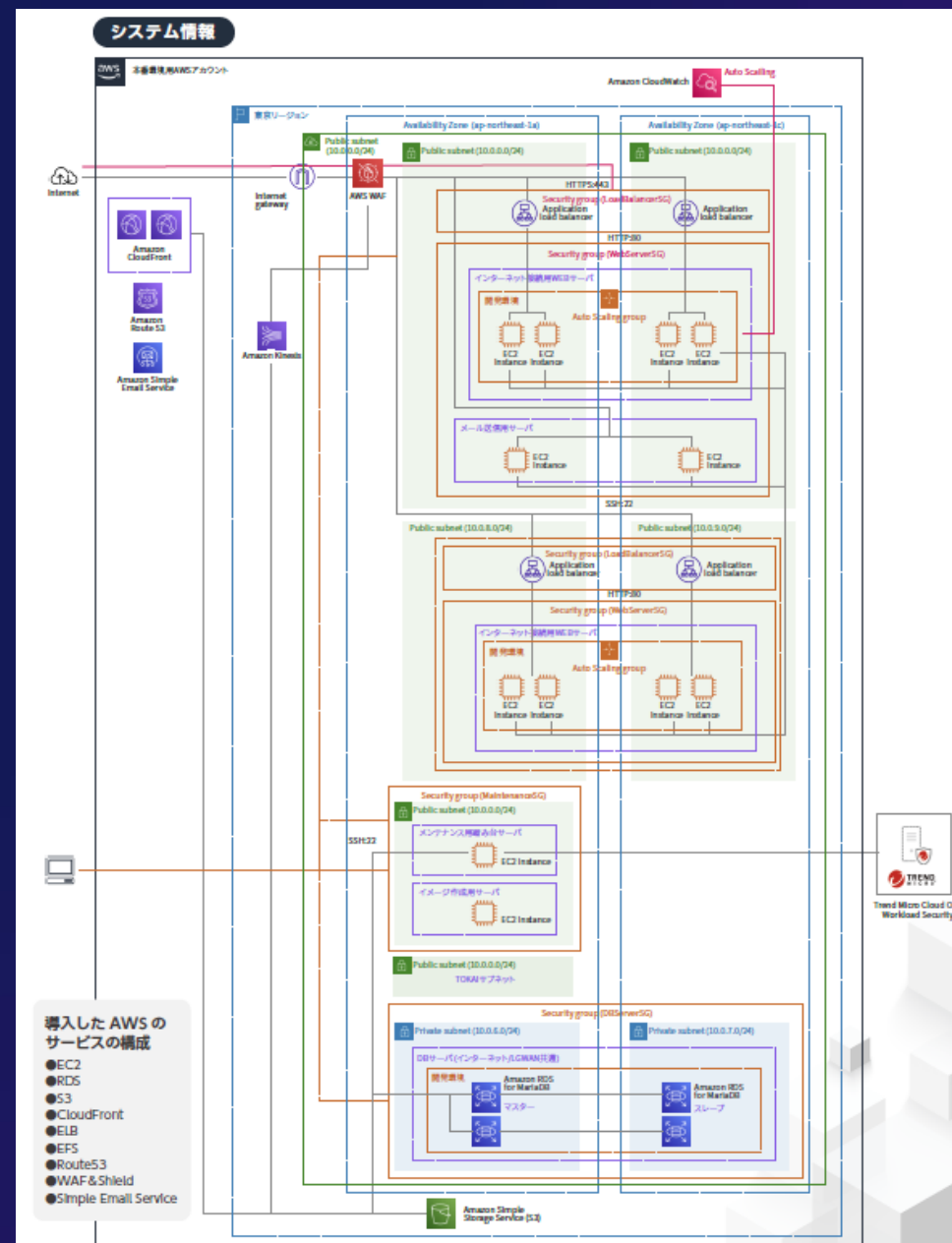
80 を超える自治体で 2,000 万人以上の予約実績

・開発要件・課題

- ワクチンの流通状況によって相当数のアクセスが想定されるが、**予測が困難**
- アクセス数が減少した際は**規模を縮小できる柔軟性**が必要

・アーキテクチャのポイント

- アクセス数による負荷状況を **Amazon CloudWatch** でモニタリングし **Auto Scaling** で柔軟に変更
- **CloudFront Functions** で先着方式と抽選方式の混合方式による **バーチャル待合室** を作成
- **最大 5 分間で 1,000 万リクエストのアクセス**でも動作するシステムを構築



大量同時アクセスを処理する上での 2 つの考え方

1. 負荷に応じてリソースをスケーリングする対策

- a. 静的コンテンツの処理をオフロード
- b. 仮想サーバを動的にスケーリング
- c. データベースの読み込み処理をスケーリング

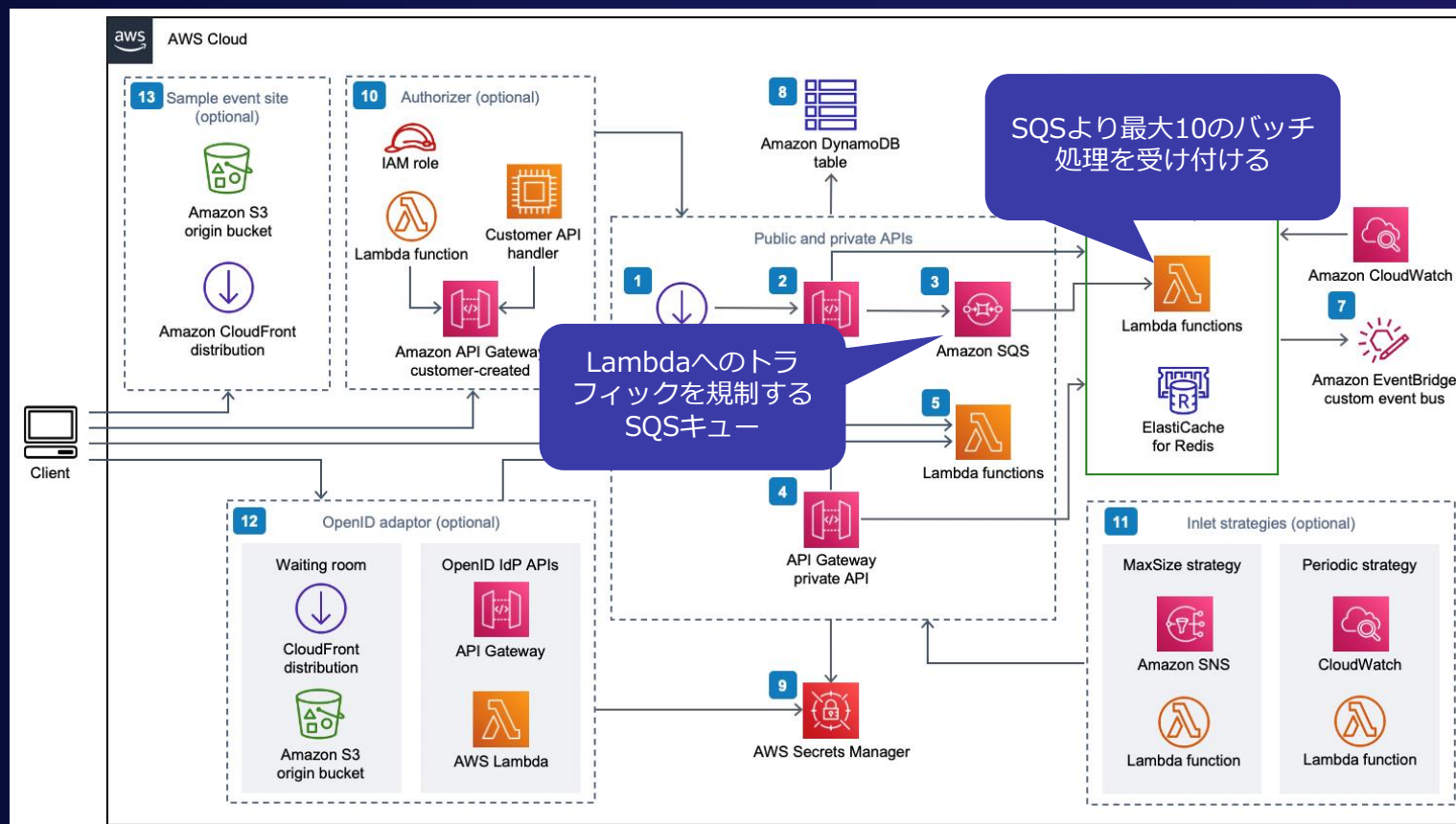
2. トラフィック量を制御する対策

- a. エッジロケーションでのフィルタリング
- b. キューを用いたバッファリング

2-b. キューを用いたバッファリング

サーバレスアーキテクチャでバーチャル待合室を構築

- Amazon API Gateway, Amazon SQS, AWS Lambda, Amazon Dynamo DB
- リクエストをキュー (Amazon SQS) を用いてバッファリングし、バックエンド (Lambda) へ流れるトラフィックを制御



AWS Virtual Waiting Room

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

<https://aws.amazon.com/jp/solutions/implementations/aws-virtual-waiting-room/>

後半のまとめ：大量同時アクセスに対応するアーキテクチャ

1. 負荷に応じてリソースをスケーリングする対策

- a. 静的コンテンツの処理をオフロード (CloudFront + S3)
- b. 仮想サーバを動的にスケーリング (CloudWatch + Auto Scaling)
- c. データベースの読み込み処理をスケーリング (Aurora リードレプリカ)

2. トラフィック量を制御する対策

- a. エッジロケーションでのフィルタリング (CloudFront Functions)
- b. キューを用いたバッファリング (AWS バーチャル待合室)

まとめ

まとめ

✓ 耐障害性を高めるアーキテクチャ

- ✓ AWS グローバルインフラストラクチャと冗長構成の考え方（シングル AZ, マルチ AZ, マルチリージョン）
- ✓ RTO と RPO を定義し、耐久性の指標を明確にする → AWS Resilience Hub の活用
- ✓ マルチサイト アクティブ-アクティブ構成 を取ることでミッションクリティカルなシステムも構築可能

✓ 大量アクセスを処理するアーキテクチャ

- ✓ 負荷に応じてリソースをスケーリングさせること、負荷の増加に備える
→ CloudFront + S3, CloudWatch + Auto Scaling, Aurora リードレプリカ
- ✓ トラフィックを制御することで、バックエンドへの負荷を吸収する
→ CloudFront Functions, バーチャル待合室

ぜひやっていただきたいこと

- 構築中・運用中のシステムを見直してみましょう
 - 耐障害性の観点
 - RTO / RPO は定義されていますか？要件は満たせていますか？
 - 大量同時アクセスの観点
 - スケーラブルになっていますか？スパイクに対応できますか？
- AWSのサービスに触ってみましょう
 - AWSリファレンスアーキテクチャ図
 - Multi-Region Application Architecture
<https://aws.amazon.com/jp/solutions/implementations/multi-region-application-architecture/>
 - AWS Hands-on for Beginners
 - スケーラブルウェブサイト構築編
https://pages.awscloud.com/event_JAPAN_Hands-on-for-Beginners-Scalable_LP.html?trk=aws_introduction_page
 - Amazon EC2 Auto Scaling スケーリング基礎編
https://pages.awscloud.com/JAPAN-event-OE-Hands-on-for-Beginners-Auto_Scaling-2021-reg-event.html?trk=aws_introduction_page

参考資料

- 事業継続性が求められる基幹システムの DR 戦略
 - <https://aws.amazon.com/jp/blogs/news/disaster-recovery-strategy-in-the-cloud/>
- AWS Resilience Hub でアプリケーションのレジリエンスを測定、改善
 - <https://aws.amazon.com/jp/blogs/news/monitor-and-improve-your-application-resiliency-with-resilience-hub/>
- 災害対策および Amazon Aurora Global Database
 - https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/aurora-global-database-disaster-recovery.html
- [AWS Black Belt Online Seminar] Amazon EC2 Auto Scaling & AWS Auto Scaling
 - https://d1.awsstatic.com/webinars/jp/pdf/services/20191002_AWS-Blakbelt_Auto_Scaling.pdf
- Visitor PrioritizationソリューションをCloudFront Functionsを使って実装するための考慮点
 - <https://aws.amazon.com/jp/blogs/news/visitor-prioritization-by-cloudfront-functions/>

Thank you!

