AWSにおける
データベース・サービスの活用

アマゾン データ サービス ジャパン株式会社
八木橋 徹平

自己紹介

amazon web services

セッションの目的

AWS上の様々なデータベース・サービスの概要と使い分を事例を交えてご紹介し、システム構築時における活用方法をご理解いただく。

amazon web services

Agenda

アジェンダ

- **データベース・サービスの概要**
- AWSのデータベース・サービス
  - Amazon RDS
  - Amazon Redshift
  - Amazon ElastiCache
  - Amazon DynamoDB
  - Databases on EC2
- まとめ

amazon web services

データベースの分類

- 多種・多用なデータベース製品

・・・ Many more

amazon web services

データベース選定の要因

- 技術要件
  - ボリューム、秒間あたりのトランザクション数、レイテンシ
  - クエリー、キー・アクセス、構造化 or 半構造化
  - 鮮度、揮発性、ACID属性
  - バックアップ、レプリケーション、高可用性（HA）
  - 管理・監視ツール、解析ツールとの連携
- ビジネス要件
  - 初期費用、メンテナンス・コスト、移行コスト
  - 社内標準（政治）
  - 過去の実績、技術者の熟練度（社内・社外）

amazon web services

SQL vs. NoSQL

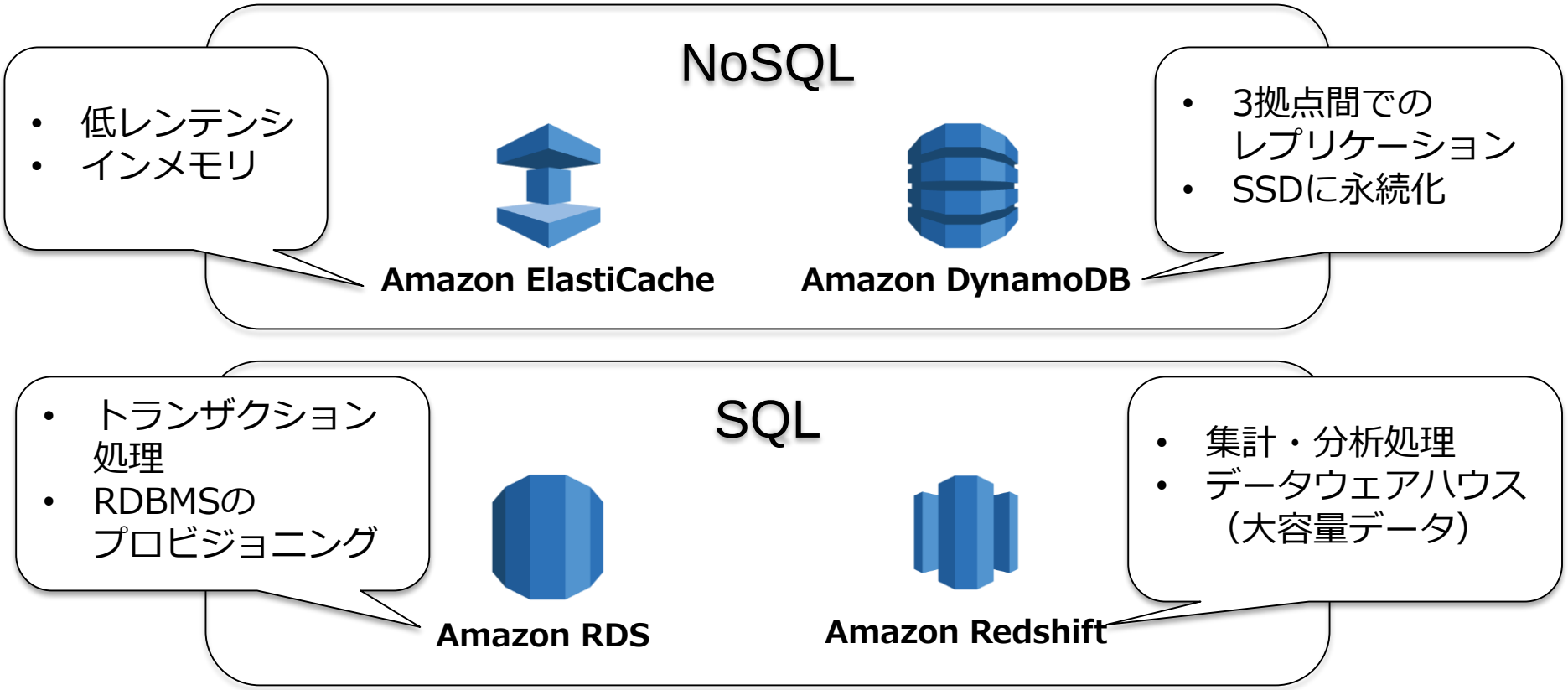
- 日本ではSQL対応のデータベースが多用されている。
  - テーブル構造に対するSQLの：
    トランザクション処理 or 分析処理（データウェアハウス）
  - ACID属性 vs. BASE属性

- NoSQLの利用価値は？
  - 低レイテンシ・高スループット、シンプルなAPI
  - Webセッション管理
  - Publish・Subscribeモデル、イベント処理
  - JSON形式データの格納
  - ソーシャル・グラフ、BOM（Bill of material）ツリーの検索 等

amazon web services

AWSデータベース・サービスの概要

- フルマネージド・データベースの特性に応じた使い分け



+ Databases on EC2

amazon web services

アジェンダ

Agenda

- データベース・サービスの概要
- **AWSのデータベース・サービス**
  - **Amazon RDS**
  - **Amazon Redshift**
  - **Amazon ElastiCache**
  - **Amazon DynamoDB**
  - **Databases on EC2**
- まとめ

amazon web services

Amazon RDS

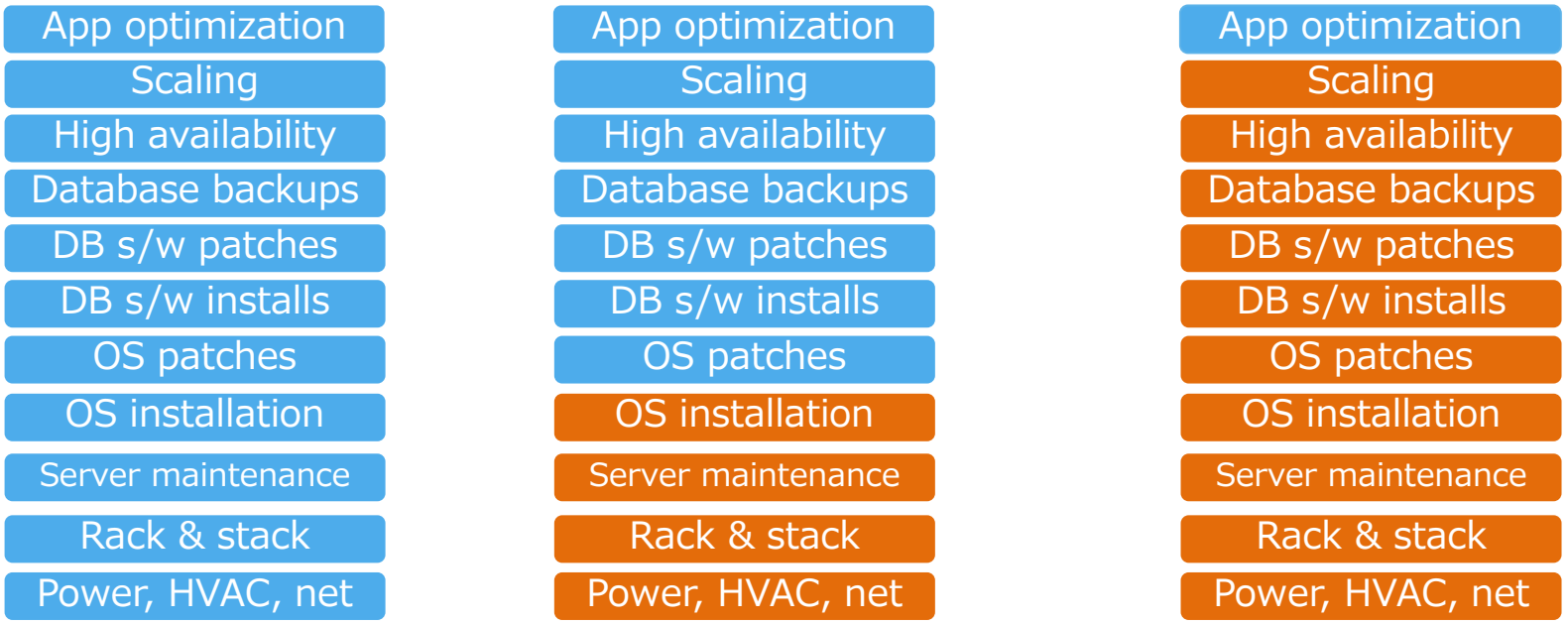
amazon web services

Amazon RDSとは？

- 構築
  - 数クリック or APIでDBサーバを操作
  - EC2同様、初期費用無し、時間単位の従量課金
- 移行
  - 4種類のエンジンをサポート
  - 既存アプリケーションの変更不要
- 運用
  - 可用性向上のための機能
  - モニタリング、障害検出/復旧、パッチ、スケーリングが容易
- セキュリティ
  - セキュリティグループ、VPC対応

amazon web services

vs. RDS



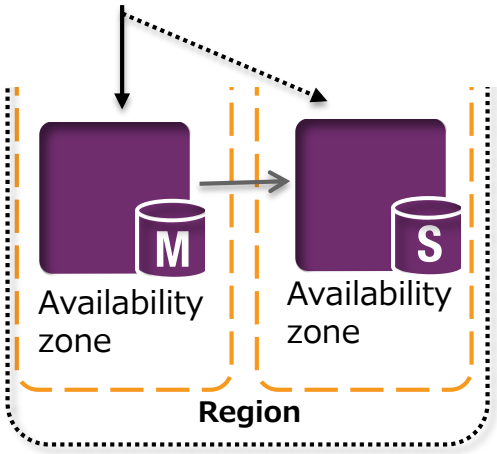
お客様がご担当する作業　　AWSが提供するマネージド機能

amazon web services

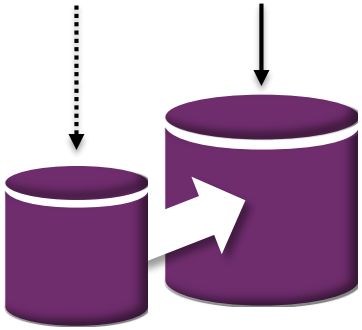
簡単に高性能・可用性の構成を実現

	可用性	スループット増	レイテンシ
スケールアップ		✓	
Multi AZ	✓		
リードレプリカ		✓	
プロビジョンド IOPS		✓	✓

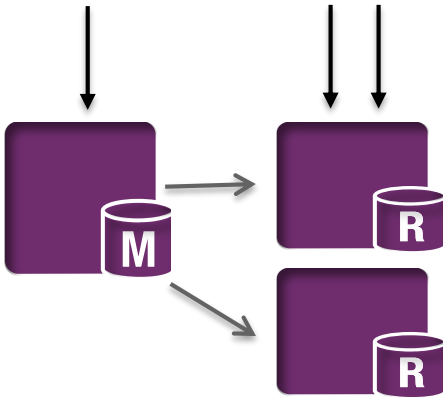
Multi-AZ



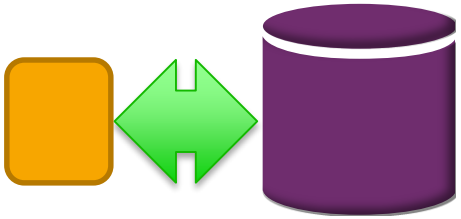
スケールアップ



リードレプリカ

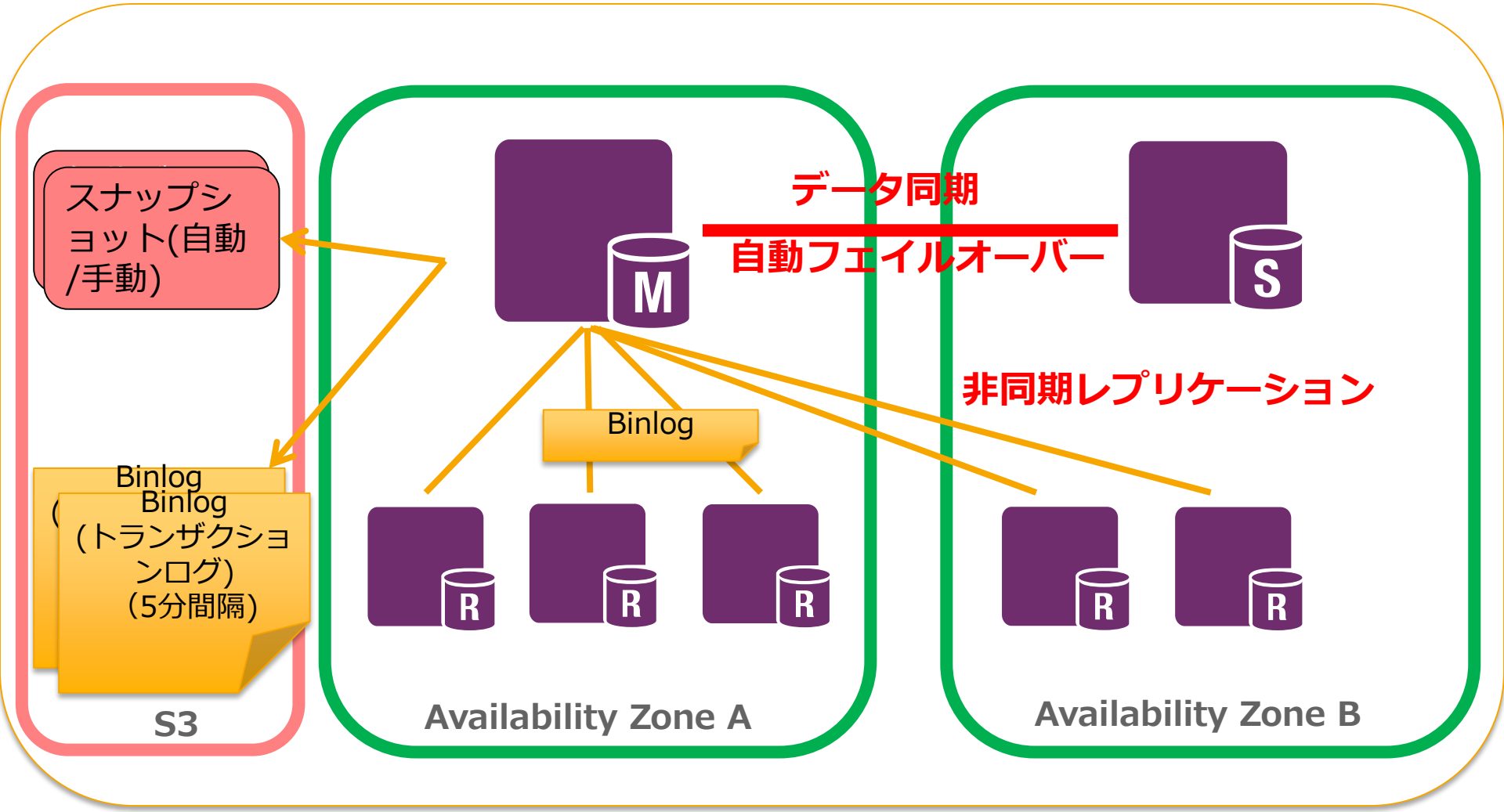


プロビジョンド IOPS



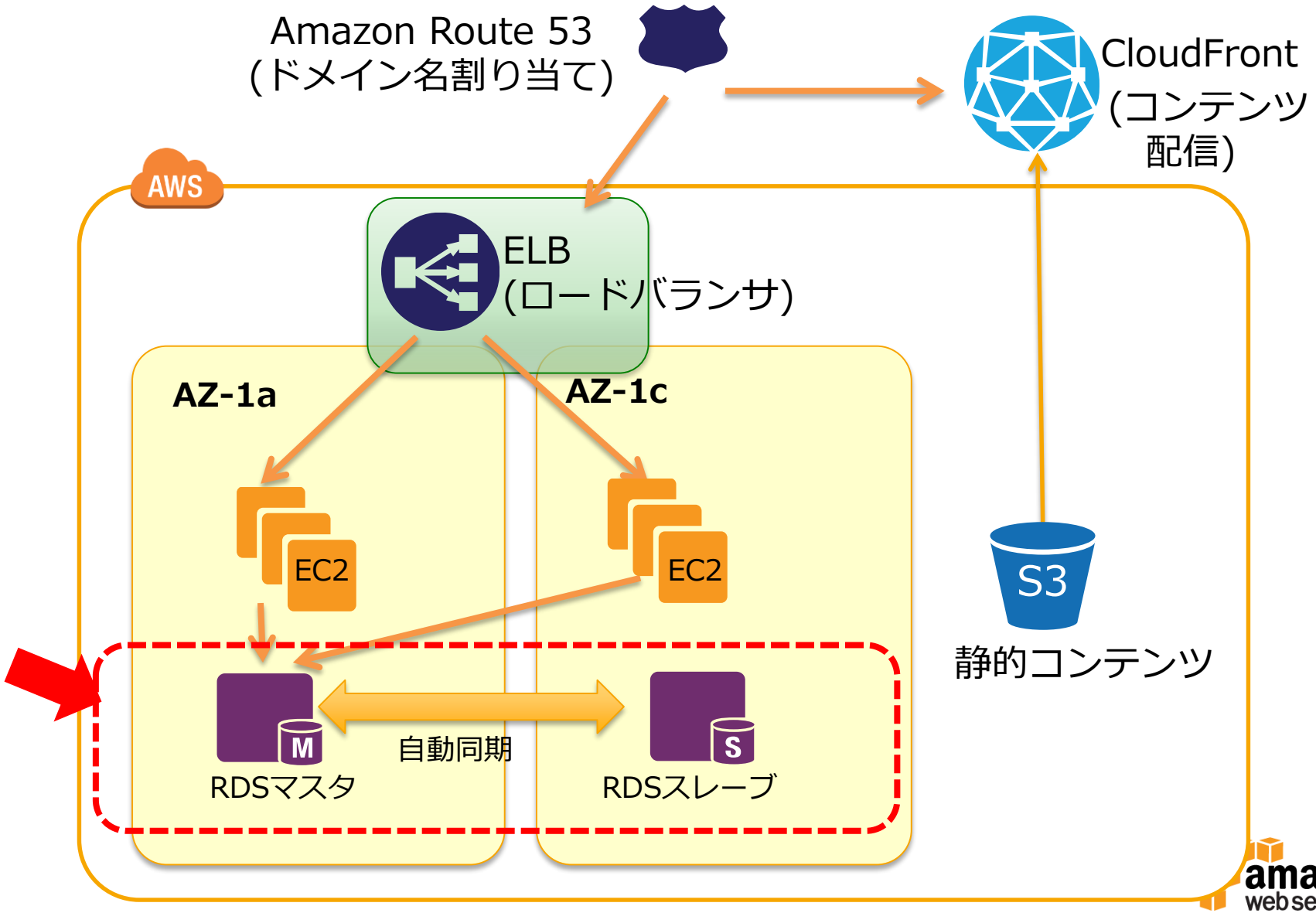
amazon web services

RDS（MySQL）アーキテクチャ



amazon web services

RDSの構成例


Amazon Route 53
(ドメイン名割り当て)
CloudFront
(コンテンツ
配信)
AWS
ELB
(ロードバランサ)
AZ-1a
AZ-1c
EC2
EC2
S3
静的コンテンツ
M
S
自動同期
RDSマスタ
RDSスレーブ

Case Study

amazon web services

Case Study

amazon web services

RDSの使いどころ

Tips

- SQLによるトランザクション処理
- 既存資産（アプリ、人材など）の活用
- 運用管理コスト（バックアップ、パッチ適用作業など）の低減
- データセンターを跨いだ容易な冗長構成（Multi-AZ）

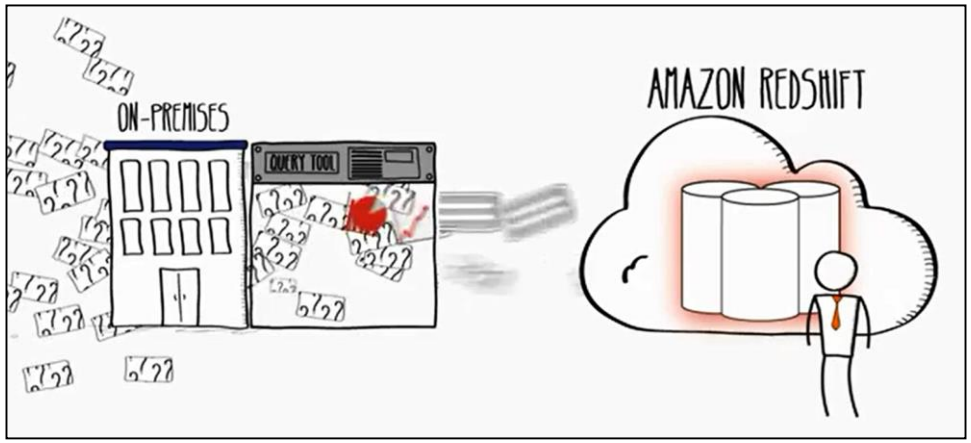
amazon web services

Amazon Redshift

amazon web services

Amazon Redshiftとは？

- Data Warehouse as a Service – 分析用の大容量統合業務データの管理サービス（フルマネージドサービス）
- 拡張性：160GB～1.6PBまで拡張可能
- 高速：カラムナ型、超並列演算(MPP)
- 低額：インスタンスの従量課金（初期費用、ライセンス費用不要）



amazon web services

列指向型データベース

- Redshiftは列指向（カラムナ）型データベース

行指向 – トランザクション処理向き

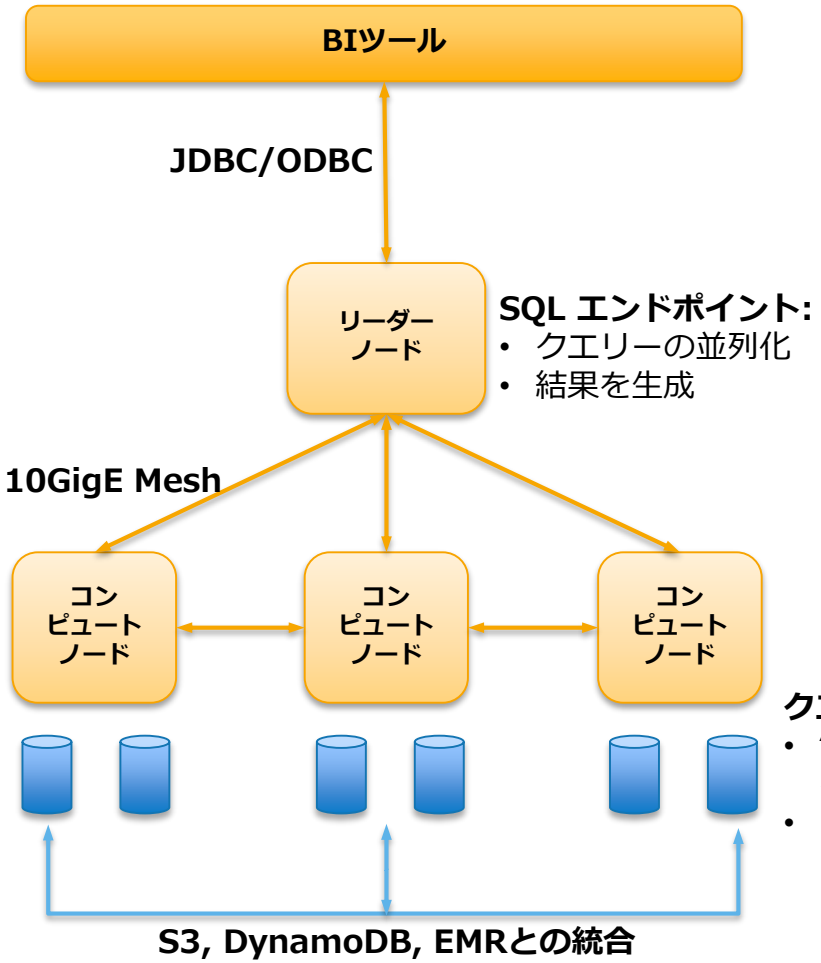
orderid	name	price
1	Book	100
2	Pen	50
	…	
n	Eraser	70

列指向 – 分析処理向き

orderid	name	price
1	Book	100
2	Pen	50
	…	
n	Eraser	70

amazon web services

Redshiftのアーキテクチャ



- リーダーノードを経由して SQLクエリーを実行
- 各コンピュートノードで演算が並列実行
- 各コンピュートノードにローカルストレージを保持
- データは、S3、DynamoDB、EMRから直接コンピュート・ノードへ並列ロード

amazon web services

ノード・タイプ

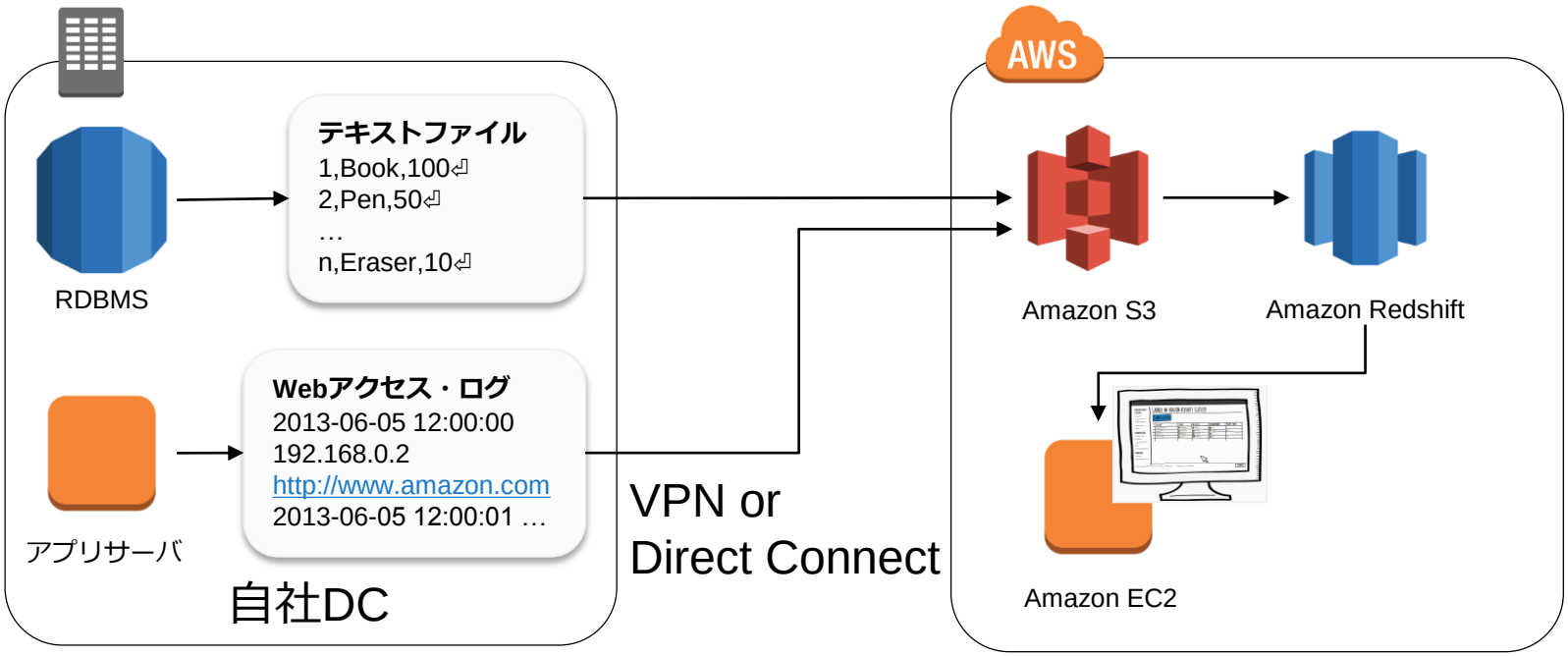
- 従量課金 = ノード数 x 時間単価
- コンピュートノードのみの課金（リーダーノードは課金対象外）

DW1 - Dense Storage						
	vCPU	ECU	Memory(GB)	Storage	I/O	Price
dw1.xlarge	2	4.4	15	2TB HDD	0.30GB/s	$1.250 1 時間あたり
dw1.8xlarge	16	35	120	16TB HDD	2.40GB/s	$10.000 1 時間あたり
DW2 - Dense Compute						
dw2.large	2	7	15	0.16TB SSD	0.20GB/s	$0.330 1 時間あたり
dw2.8xlarge	32	104	244	2.56TB SSD	3.70GB/s	$6.400 1 時間あたり

amazon web services

Redshiftの構成例

- オンプレ環境との連携



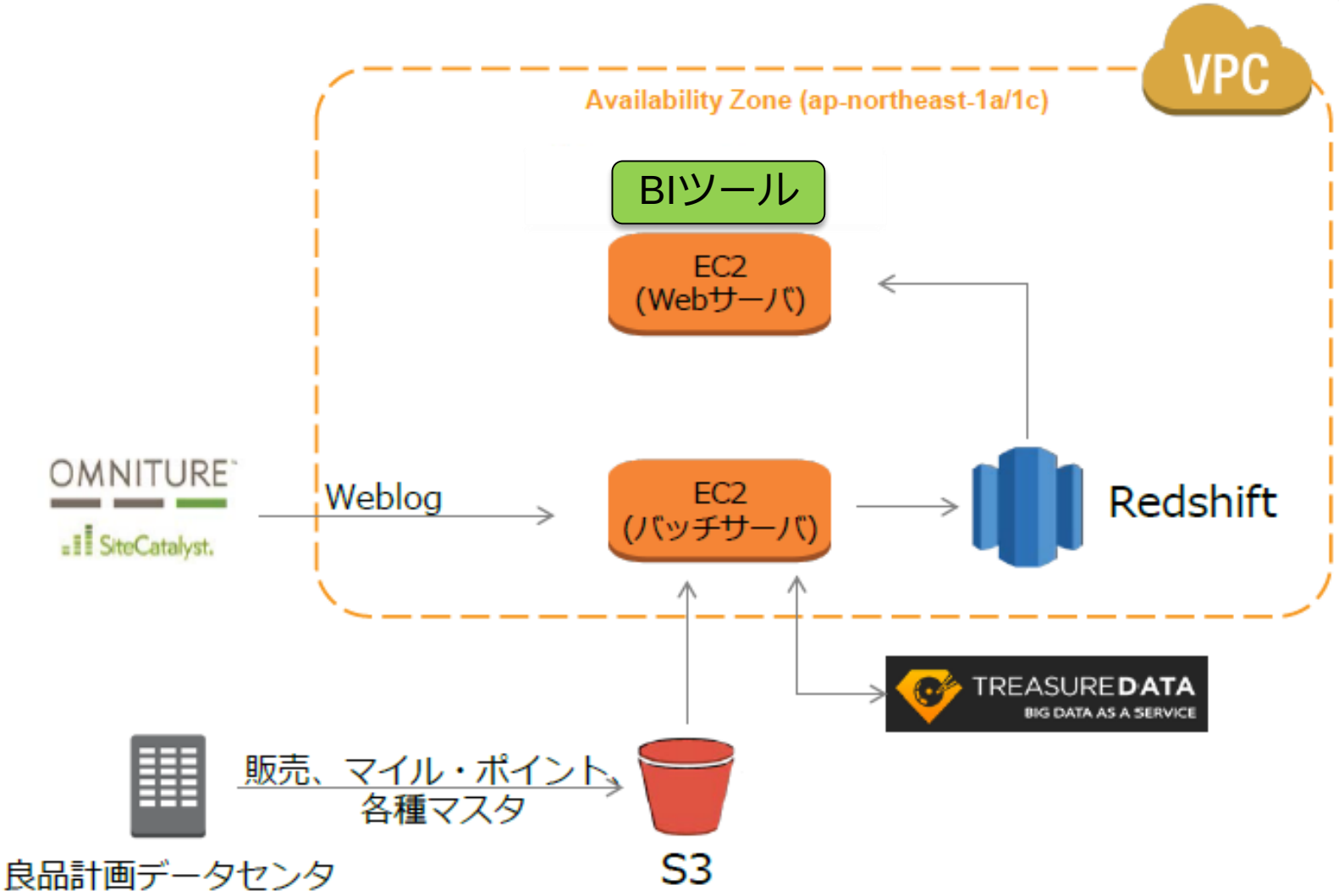
Case Study

MUJI Passport の導入目的

- ネット・リアルの区別なく無印良品のファンの方とコミュニケーションを図る
  - 複数メディアを跨ったデータの収集・解析
  - **ソーシャル・メディア、実店舗、インターネット**
- 持続的な来店客数増 -> 売上の向上
- 値引きの最小化 -> **ターゲット・マーケティング**

amazon web services

分析関連のシステム構成



amazon web services

Redshiftの使いどころ

Tips

- いわゆるデータウェアハウスとしての利用（データ容量 数百GB以上）
- OLTP用途では使用しない
- Business Intelligence（BI）ツールからの分析
- 投資対効果（ROI）が不透明な中で、大規模な投資リスクを避ける

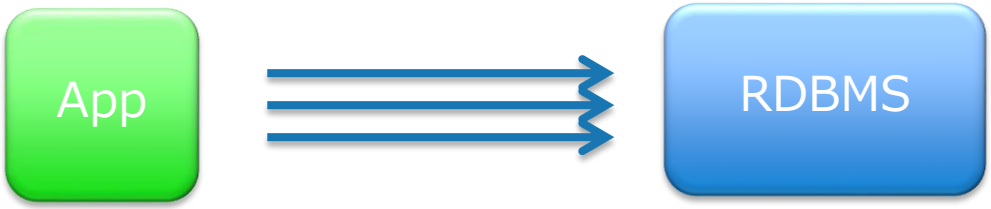
amazon web services

Amazon ElastiCache

amazon web services

メモリキャッシュとは？（１）

- Web+DBアプリの典型的な構成

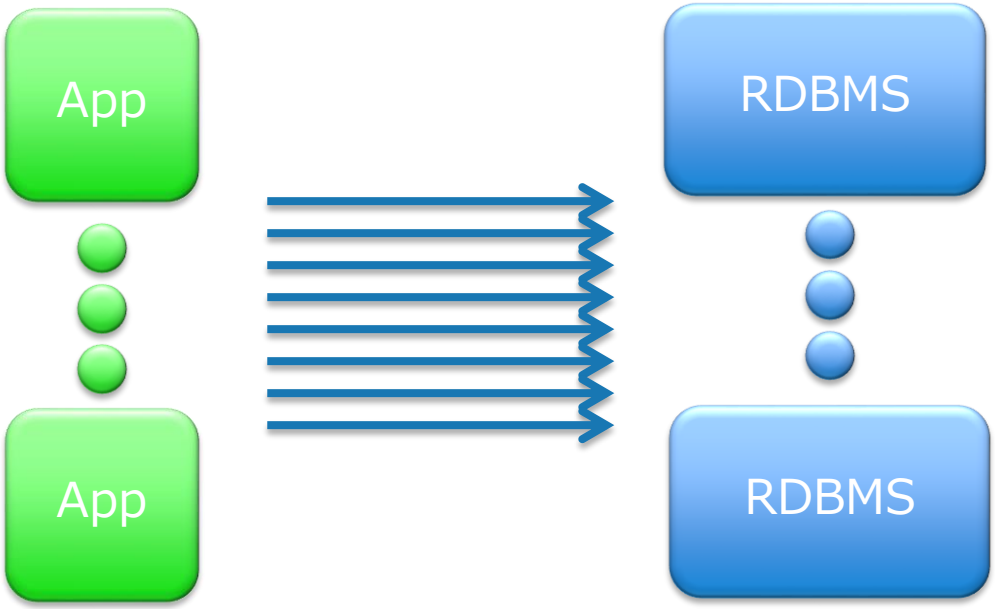


1. クライアントからのリクエスト
2. Appサーバが、DBサーバに問い合わせ
3. DBサーバが結果を戻す
4. Appサーバがレスポンスをクライアントに返す

amazon web services

メモリキャッシュとは？（2）

- トラフィックが増えると

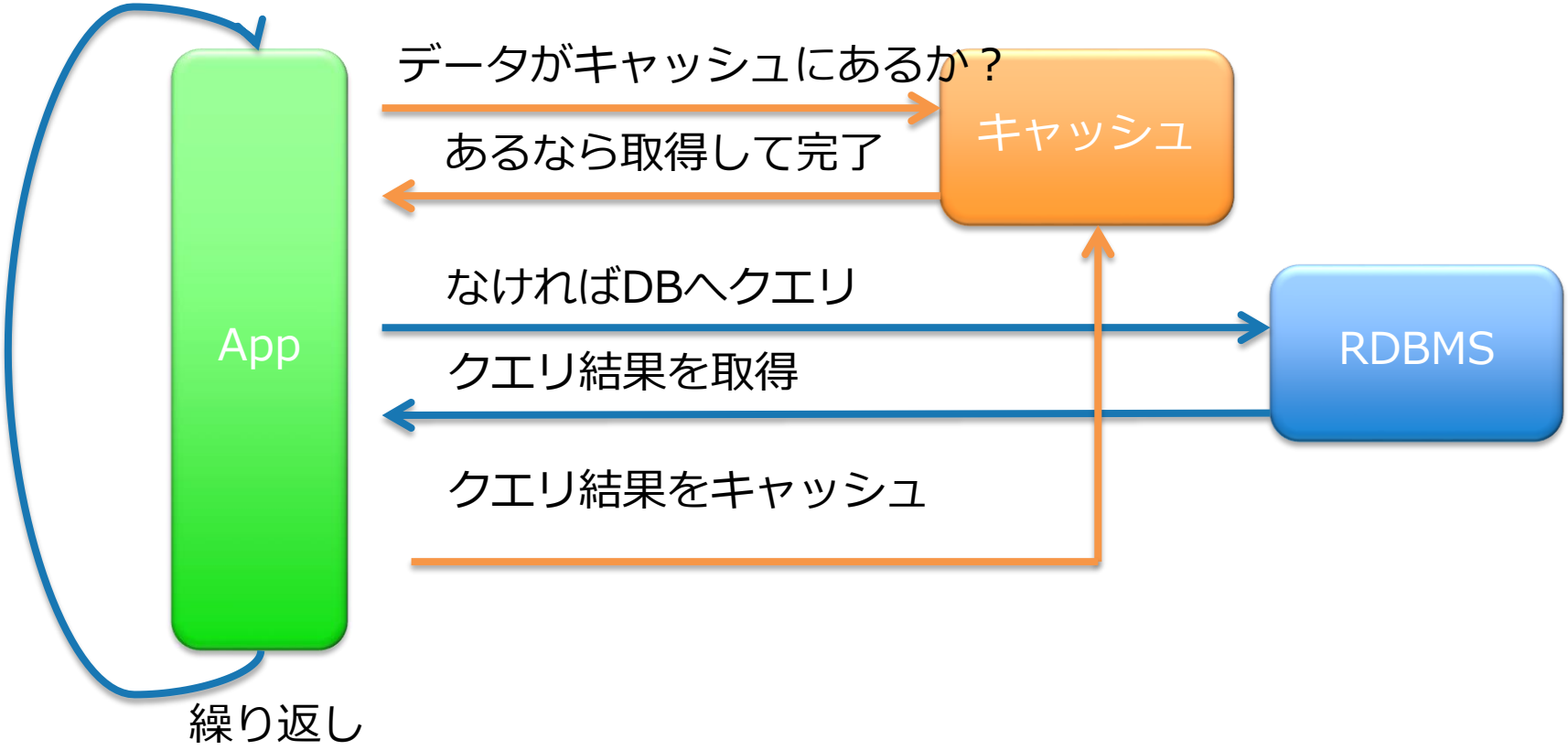


5. Appサーバ,DBサーバをスケール
6. 効果・効率・コスト的な面、DBをスケールさせる難易度は？

amazon web services

メモリキャッシュとは？（３）

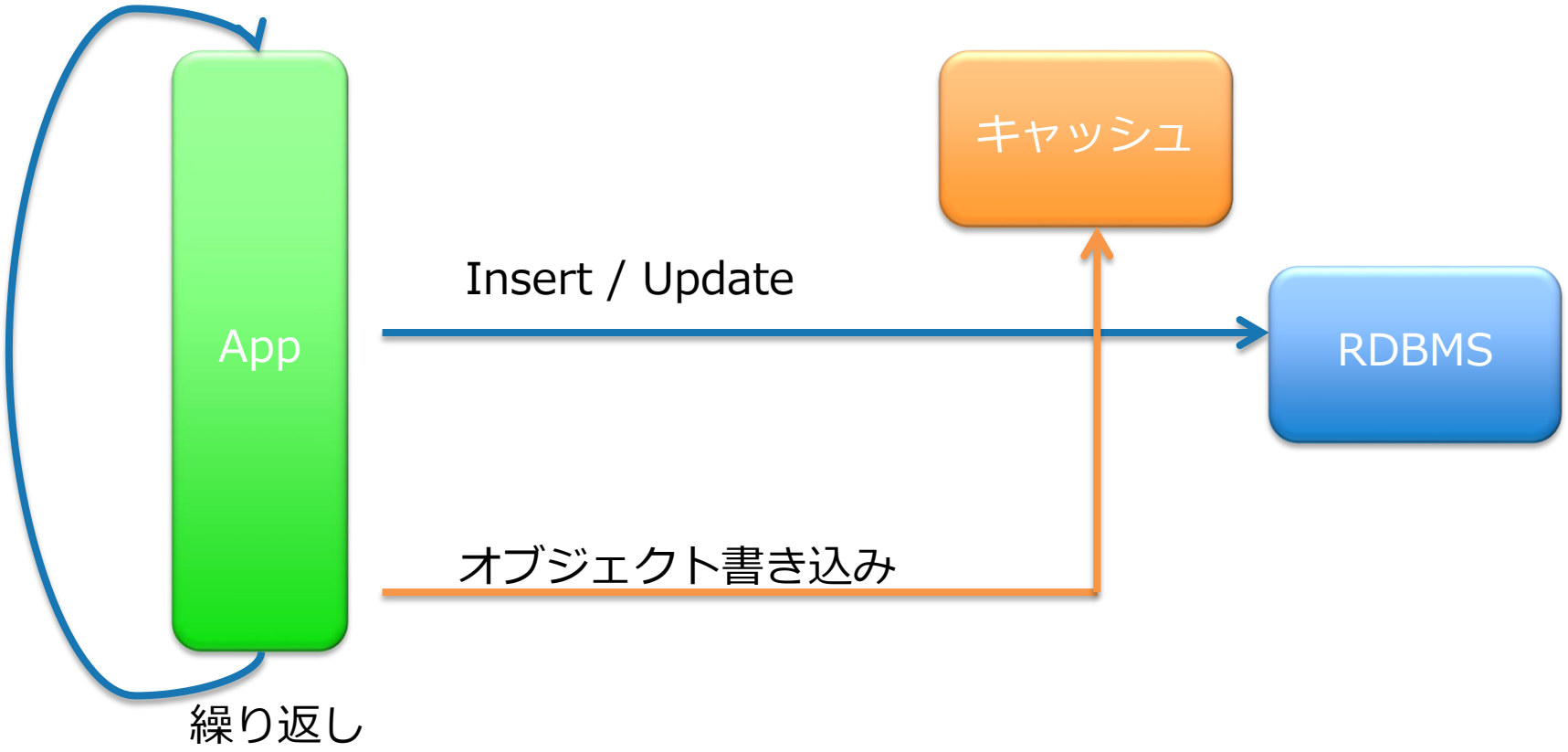
- データ参照時の操作



amazon web services

メモリキャッシュとは？（４）

- 更新時の操作



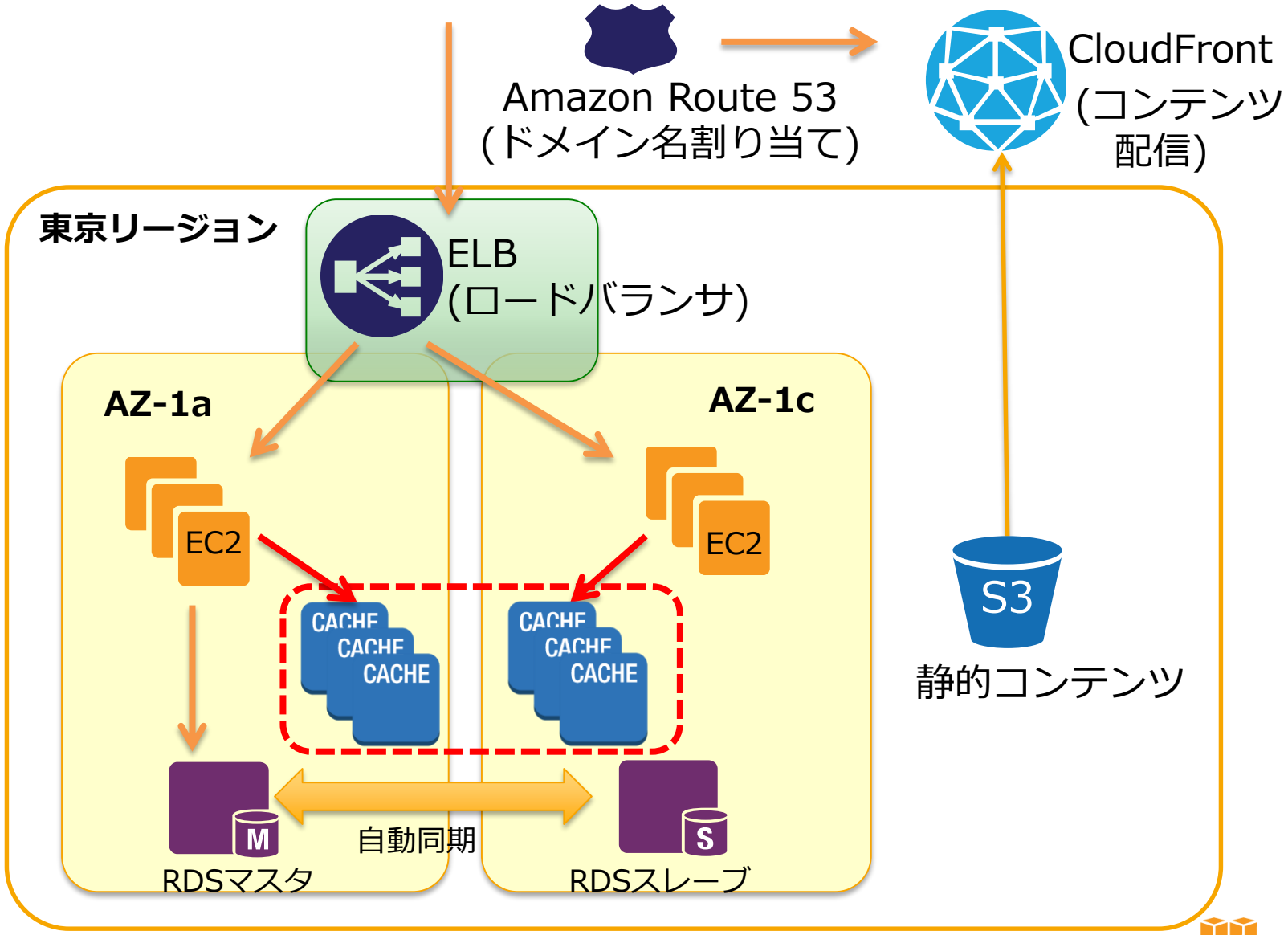
amazon web services

Amazon ElastiCacheとは

- 構築
  - キャッシュクラスタを数クリックで起動
  - EC2同様、初期費用無し、時間単位の従量課金
- 移行
  - 2種類のエンジン(memcached, redis)をサポート
  - 既存アプリケーションの変更不要
- 運用
  - 可用性を向上させる機能
  - モニタリング、自動障害検出、復旧、拡張、パッチ管理機能を提供
- セキュリティ
  - セキュリティグループ、VPC対応

amazon web services

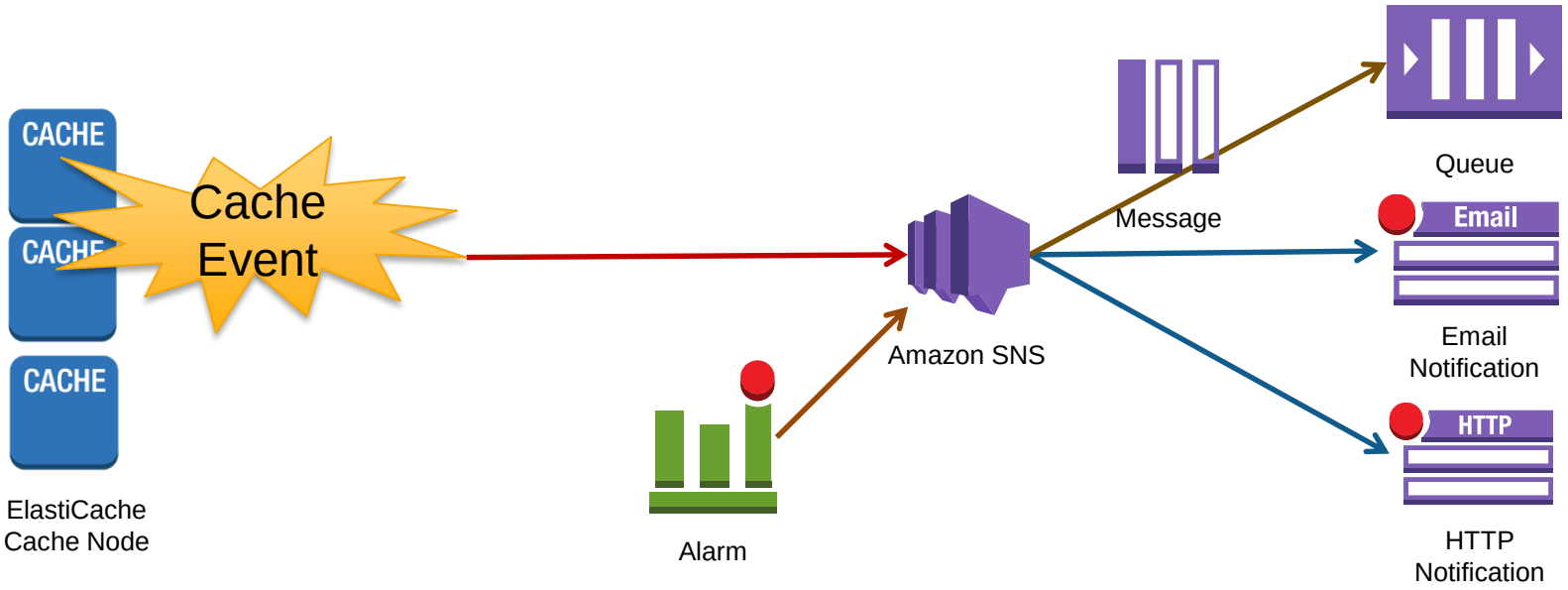
典型的なWeb＋Cacheアーキテクチャ



amazon web services

イベント通知 (Cache Event)

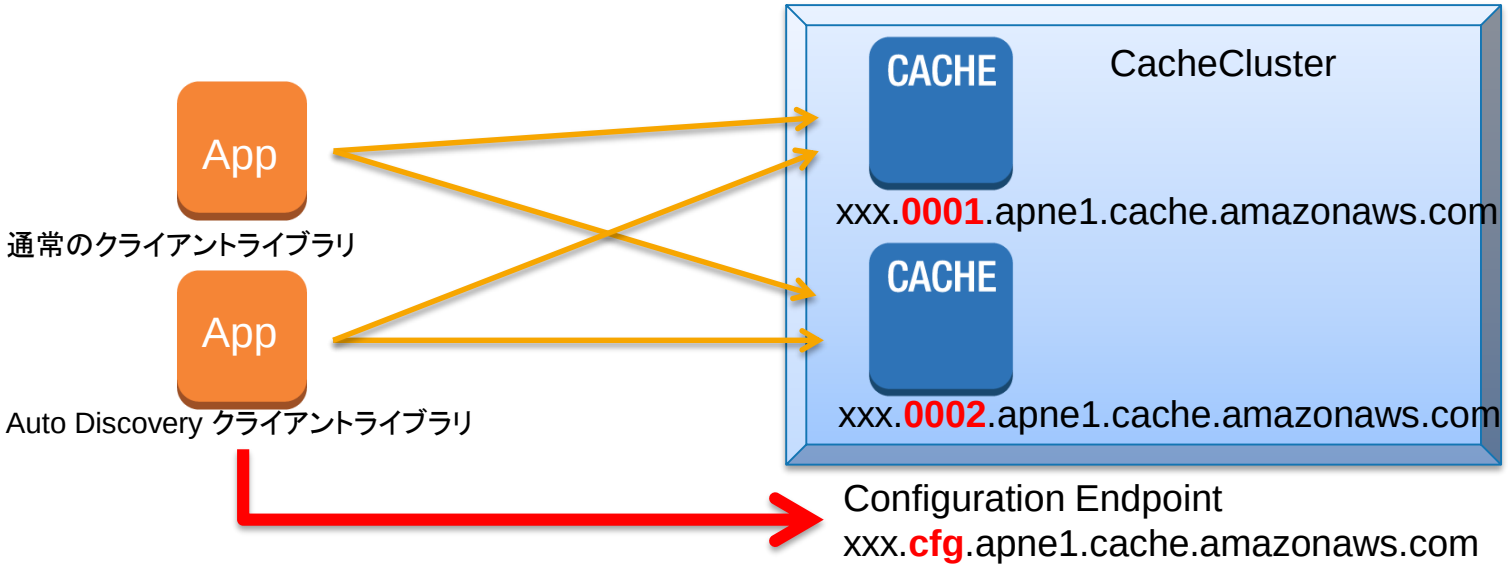
- ElastiCacheで発生した10以上のイベントをSNS経由でPush通知
  - 再起動、ノード追加、ノードリプレイス、設定変更、メンテナンス終了、etc
- 監視システムと組み合わせることで運用自動化が容易に



amazon web services

Auto Discovery for memcached

- 従来のクライアント側の設定
  - Cache Clusterの全エンドポイントを接続先として設定する。
- Auto Discoveryクライアント(Java, PHP)
  - Cache ClusterのConfiguration Endpointを接続先として設定すると、全エンドポイントを自動取得・設定し、接続する
- 注意
  - Configuration Endpointは、Cache Clusterの ロードバランサー(Proxy)ではなく、あくまでもメタデータを取得するEndpointとなる



http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/AutoDiscovery.html

amazon web services

ElastiCacheの使いどころ

Tips

- キー・ベースでのアクセス
  - set（key, value）、value = get（key）
- あくまでキャッシュであり、主はRDBMS等に永続化されることを前提とする
- 更新頻度が低く、アクセス頻度が高いデータをキャッシュ
  => キャッシュ・ヒット率が高い

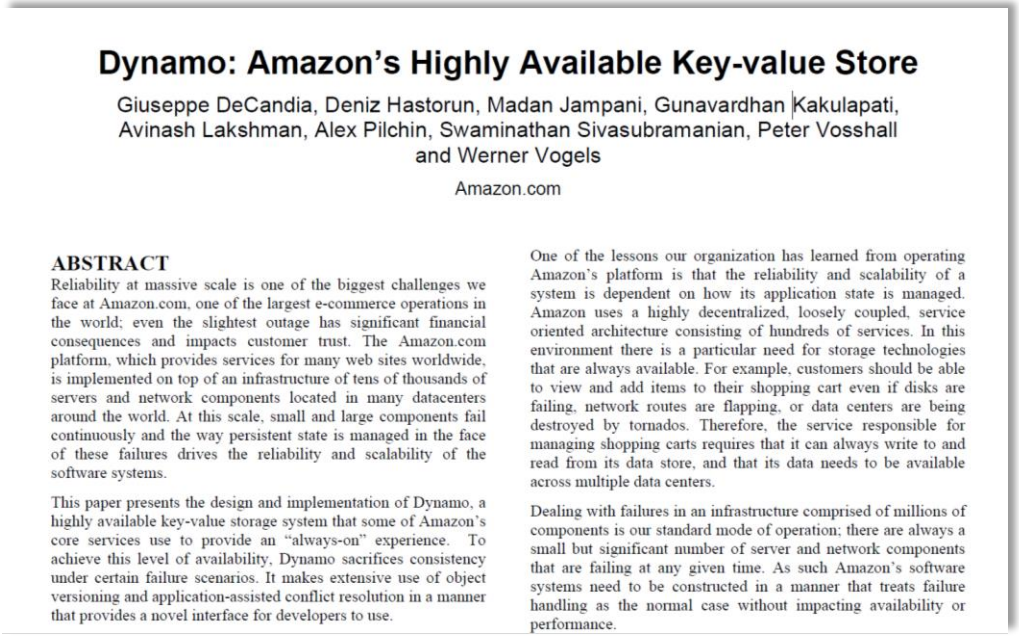
amazon web services

Amazon DynamoDB

amazon web services

Amazon DynamoDBとは？

- NoSQL as a Service
- 高速・一貫したパフォーマンスを維持
- シームレスなスケーラビリティ、低コスト



Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels

Amazon.com

ABSTRACT

Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters around the world. At this scale, small and large components fail continuously and the way persistent state is managed in the face of these failures drives the reliability and scalability of the software systems.

This paper presents the design and implementation of Dynamo, a highly available key-value storage system that some of Amazon's core services use to provide an "always-on" experience. To achieve this level of availability, Dynamo sacrifices consistency under certain failure scenarios. It makes extensive use of object versioning and application-assisted conflict resolution in a manner that provides a novel interface for developers to use.

One of the lessons our organization has learned from operating Amazon's platform is that the reliability and scalability of a system is dependent on how its application state is managed. Amazon uses a highly decentralized, loosely coupled, service oriented architecture consisting of hundreds of services. In this environment there is a particular need for storage technologies that are always available. For example, customers should be able to view and add items to their shopping cart even if disks are failing, network routes are flapping, or data centers are being destroyed by tornados. Therefore, the service responsible for managing shopping carts requires that it can always write to and read from its data store, and that its data needs to be available across multiple data centers.

Dealing with failures in an infrastructure comprised of millions of components is our standard mode of operation; there are always a small but significant number of server and network components that are failing at any given time. As such Amazon's software systems need to be constructed in a manner that treats failure handling as the normal case without impacting availability or performance.

運用管理必要なし

低レイテンシ、SSD

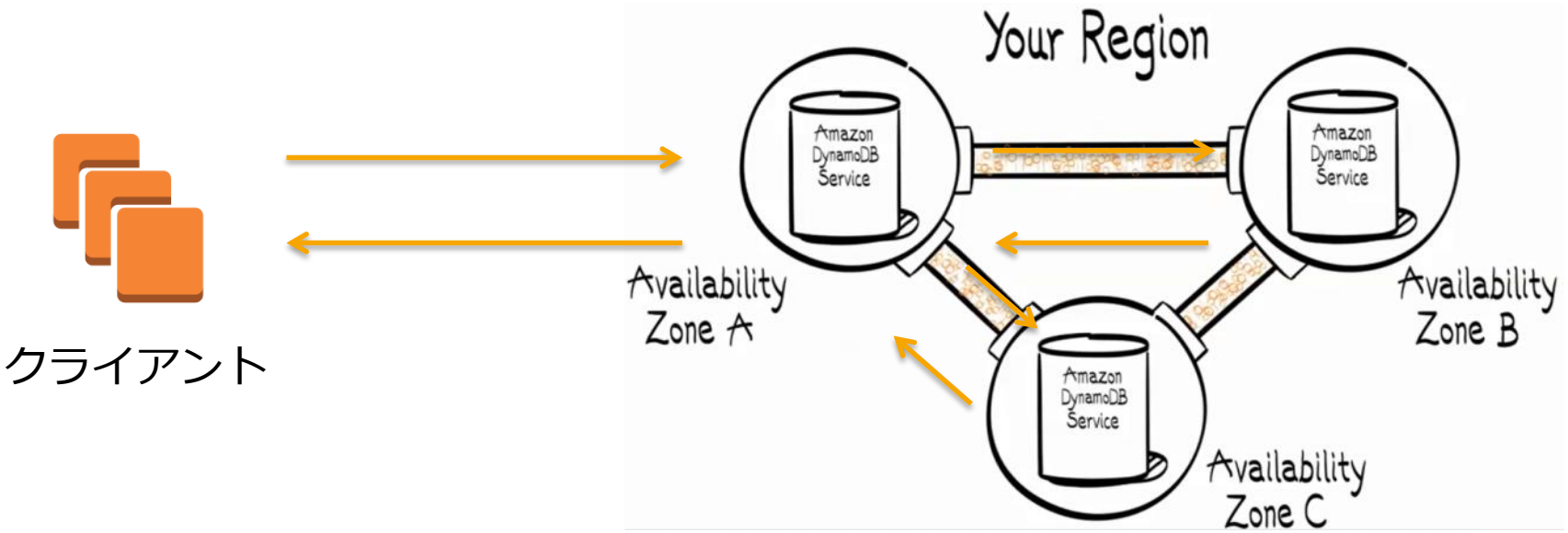
プロビジョンスループット

無限に使えるストレージ

amazon web services

特徴１：管理不要で高信頼性

- SPOFの存在しない構成
- データは3箇所のAZに保存されるので信頼性が高い
- ストレージは必要に応じて自動的にパーティショニングされる



amazon web services

特徴２：プロビジョンスループット

- ReadとWrite、それぞれに対して必要な分だけのスループットキャパシティをプロビジョンする（割り当てる）ことができる
- 例えば一般的なReadヘビーなDBなら
  - Read：1,000
  - Write：100
- ライトヘビーなDBなら
  - Read：500
  - Write：500
- この値はDB運用中にオンラインで変更可能
  - ただし、スケールダウンに関しては日に４回までしかできないので注意

amazon web services

特徴３：ストレージの容量制限がない

- 使った分だけの従量課金制のストレージ
- データ容量が増えてきたのでディスクを足したり、ノードを足したりという作業が不要

amazon web services

DynamoDBを使い始めるには

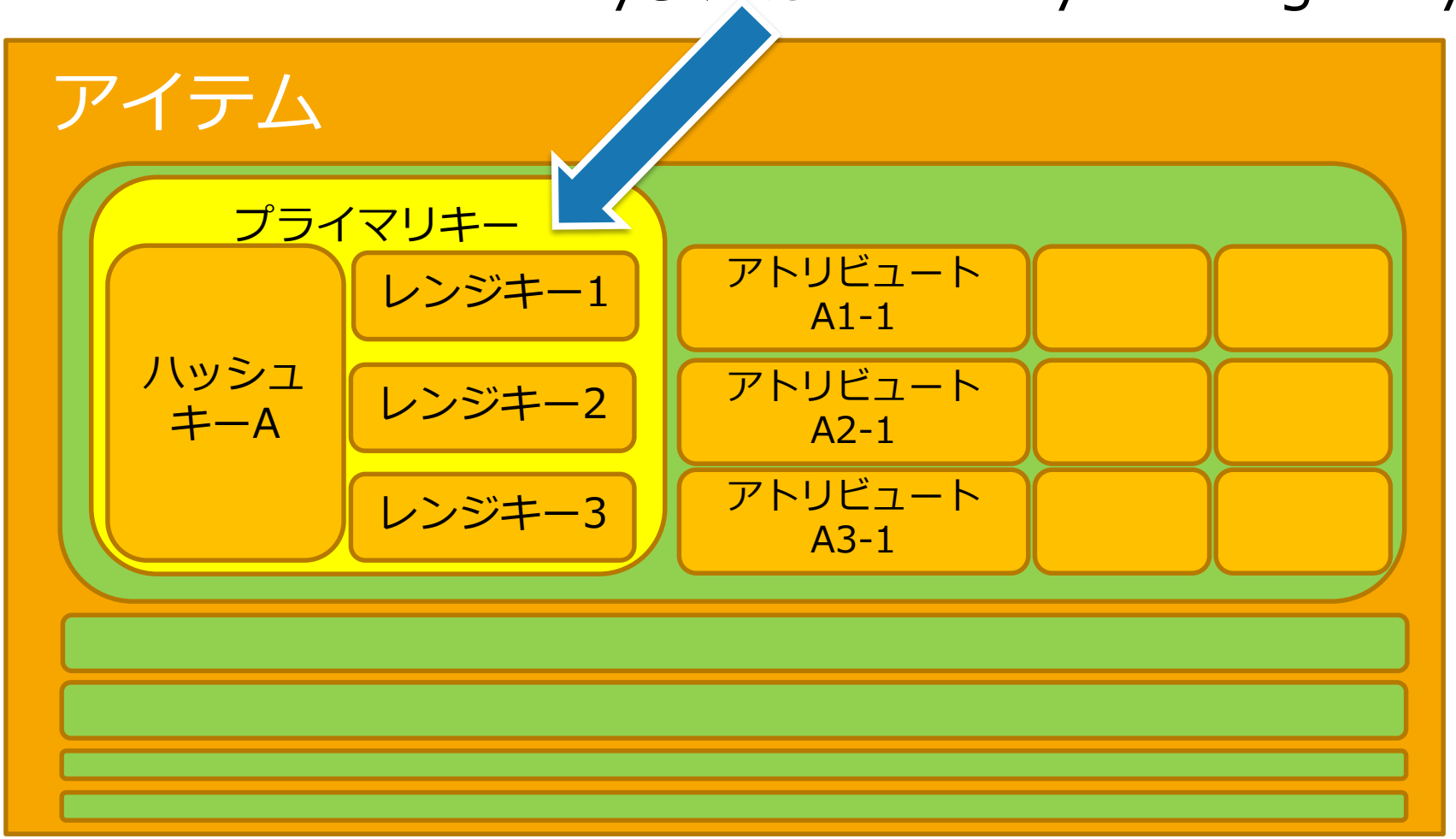
1. テーブルのKeyやIndexを決める
2. Read/Writeそれぞれのスループットを決める

That’s it, write your code!

amazon web services

DynamoDBのデータモデル

Hash keyまたは Hash key & Range key



amazon
web services

Hash Key＋Range Keyの例

- ユーザーの行動ログの蓄積

AudienceId (Hash key)	Timestamp (Range key)	Action (Action-Index)	Url	・・・
1	2013-10-01 00:01:01	Login	・・・	・・・
2	2013-10-01 00:02:02	Login	・・・	・・・
1	2013-10-01 00:21:00	Login	・・・	・・・
1	2013-10-01 00:42:00	ViewHoge		
1	2013-10-01 00:56:22	PostHoge		

amazon web services

DynamoDBの使いどころ

Tips

- キー+クエリーでのアクセス
- ３拠点でデータが保全されることによる堅牢なシステムの構築
- スループットの容易な増減によるピーク負荷への対応

amazon web services

Databases on EC2

amazon web services

Database on EC2

- EC2上にデータベース・ソフトをインストールし、自ら運用管理を行う
- マネージド・データベースのメリットは、享受できないが、特殊な要件の場合に検討
  - RDSで制限されているStored Procedureを使いたい
  - AWSで提供されていないNoSQLを利用したい
  - データベースのOSにアクセスし、ローカル・アクセスでバッチ処理を実行したい 等々
- 運用管理コストが上がる点も考慮する

amazon web services

アジェンダ

Agenda

- データベース・サービスの概要
- AWSのデータベース・サービス
  - Amazon RDS
  - Amazon Redshift
  - Amazon ElastiCache
  - Amazon DynamoDB
  - Databases on EC2
- **まとめ**

amazon web services

まとめ

- データの特性に合わせたデータベースの選択
  - スケールアップに頼らずに低コストで実現
- SQLだけではなく、NoSQL系データベースの活用
  - 処理パターンによってはRDBMSよりも適しているケースも
- AWSのマネージド・サービスでカバーしきれないシナリオの場合、Databases on EC2を活用
  - バックアップやパッチ適用作業等も考慮

amazon web services

ご清聴ありがとうございました。

amazon web services