Amazon Aurora MySQL/PostgreSQL
独自の機能のご紹介

アマゾン ウェブ サービス ジャパン株式会社
データベース スペシャリスト ソリューション アーキテクト
齋藤 航

内容についての注意点

本資料では2020年8月1日時点のサービス内容および価格についてご説明しています。
最新の情報はAWS公式ウェブサイト (http://aws.amazon.com/) にてご確認ください。

- 資料作成には十分注意しておりますが、資料内の価格とAWS公式ウェブサイト記載の価格に相違があった場合、AWS公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様がご利用される場合、別途消費税をご請求させていただきます

AWS does not offer binding price quotes. AWS pricing is publicly available and is subject to change in accordance with the AWS Customer Agreement available at http://aws.amazon.com/agreement/. Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on certain information that you have provided. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

本セッションについて

対象者

- Amazon Aurora の持つ機能にご興味をお持ちの方

ねらい

- Amazon Aurora MySQL/PostgreSQL 互換エディションが持つ機能について、それらの解決した課題と、ユースケースをご理解いただく

前提知識

- Amazon Aurora の基礎知識

Amazon Aurora おさらい

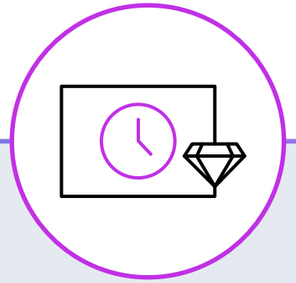
Amazon Aurora

クラウド向けに再設計された MySQL, PostgreSQL と互換性のある 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

優れた性能と拡張性

標準的なMySQLと比べて5倍、
標準的なPostgreSQLと比べて
3倍のスループットを実現;
リードレプリカを最大 15 個
追加してスケールアウト可能

高可用性と耐久性



耐障害性、自己修復機能を
兼ね備えたストレージ;
3 つのAZにわたり、6 個のコ
ピーを保持; Amazon S3への
継続的なバックアップ

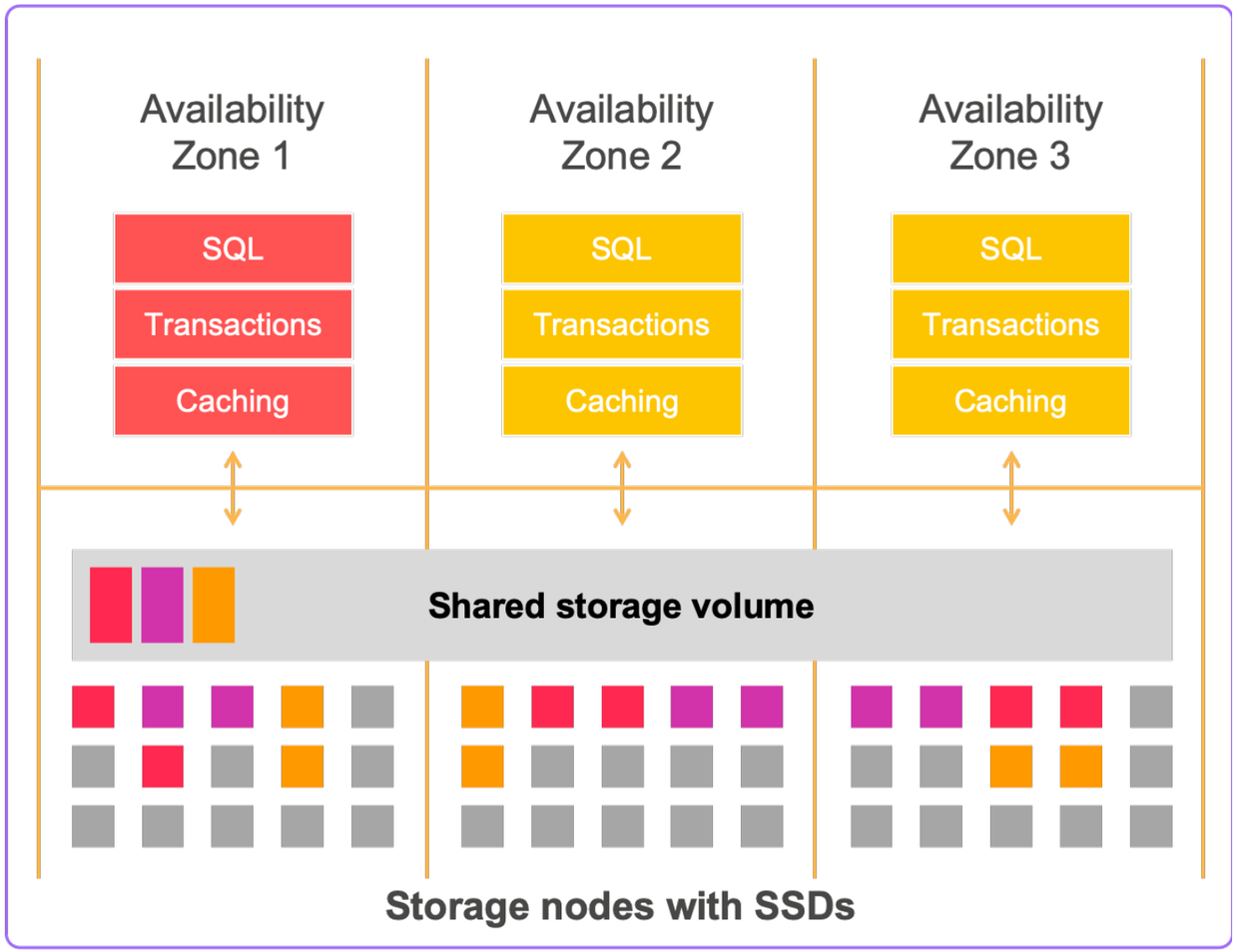
高い安全性

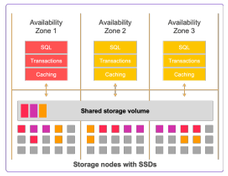
ネットワーク分離、
保管時/通信の暗号化

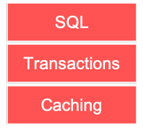
フルマネージド

ハードウェアのプロビジョニ
ング、ソフトウェアのパッチ
適用、セットアップ、構成、
バックアップといった
管理タスクからの解放

Aurora を構成するコンポーネント





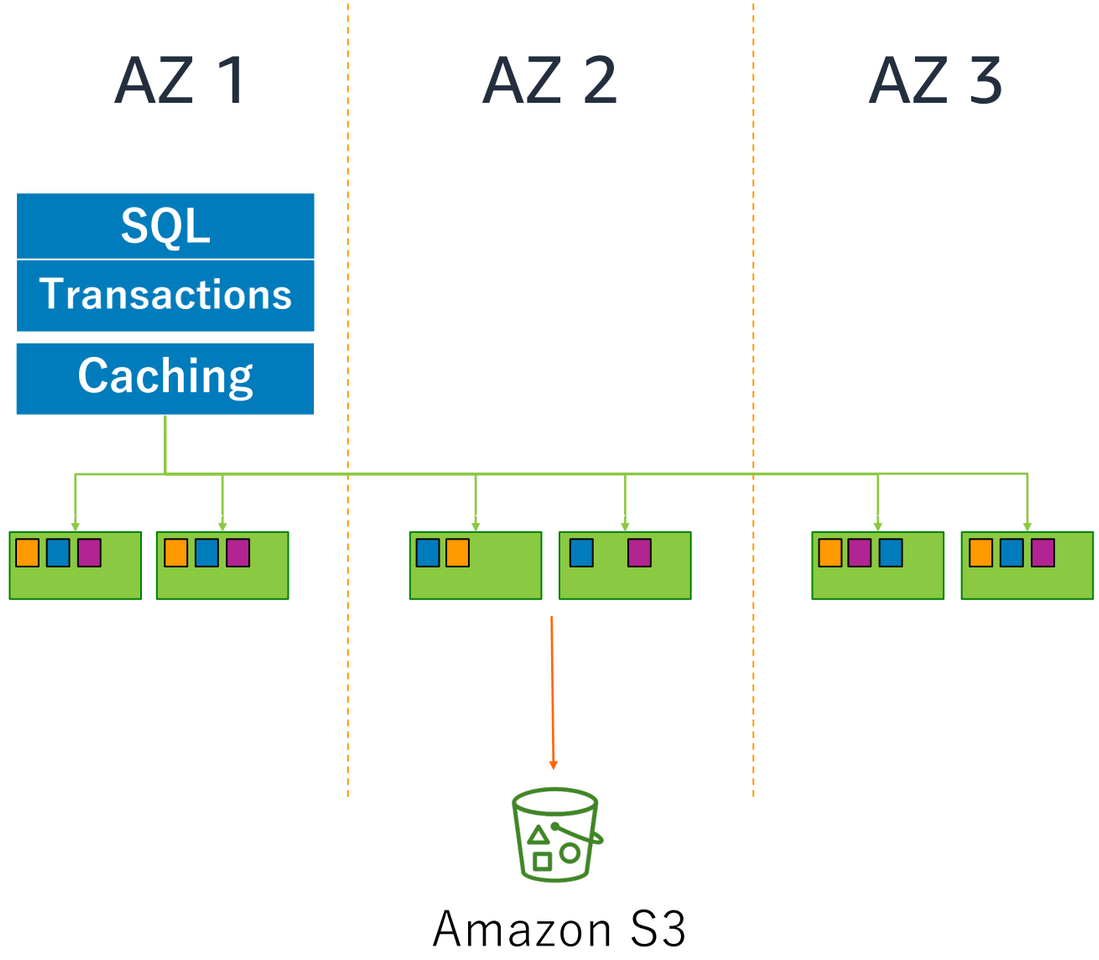


- **Amazon Aurora DB クラスター**
  - Amazon Aurora の管理単位
  - Writer インスタンス、Readerインスタンス、クラスターボリュームの総称
- **Writer インスタンス (プライマリ)**
  - 読み込み、書き込みを行うインスタンス
- **Reader インスタンス**
  - 読み込み専用のインスタンス (15台まで作成可能)
- **クラスターボリューム**
  - 3つの AZ 間でレプリケートされる仮想ボリューム
  - Writer インスタンスも Reader インスタンスも同じクラスターボリュームを利用
- **Aurora エンドポイント**
  - Aurora の接続先を示す URL

詳細：http://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/Aurora.Overview.html

Aurora のストレージ

- **SSDを利用したシームレスに スケールする分散ストレージ**
  - 10GB から 64TB までシームレスに自動でスケールアップ
  - 実際に使った分だけ課金
- **標準で高可用性を実現**
  - 3AZ に6つのデータのコピーを作成
  - クォーラムシステムの採用
  - 継続的に S3 へ増分バックアップ



Aurora 独自の機能の紹介

Aurora の持つ独自の機能 1/2

カテゴリ	Aurora MySQL/PostgreSQL の機能
ストレージ	データベース用に設計された log structured 分散ストレージ 3AZ にまたがる6つのデータコピー
可用性	通常 30 秒以内のフェイルオーバー 存続可能なキャッシュ オートスケール可能な最大15台のリードレプリカ Aurora グローバルデータベース (PostgreSQL は 10.11, 10.12, 11.7 互換 以降のみ)
パフォーマンス	MySQL の最大5倍, PostgreSQL の最大3倍のスループット
運用性	高速なクローン作成 継続的なバックアップ Aurora Serverless
セキュリティ	Database Activity Streams (MySQL 5.7 互換, PostgreSQL 10.7 互換 以降のみ)
AWS サービスとのインテグレーション	AWS DMS, Amazon S3, Amazon CloudWatch Logs, AWS Secrets Manager, Amazon Comprehend, Amazon SageMaker

※ オレンジ色のものが、セッション中でご紹介するもの

Aurora の持つ独自の機能 2/2

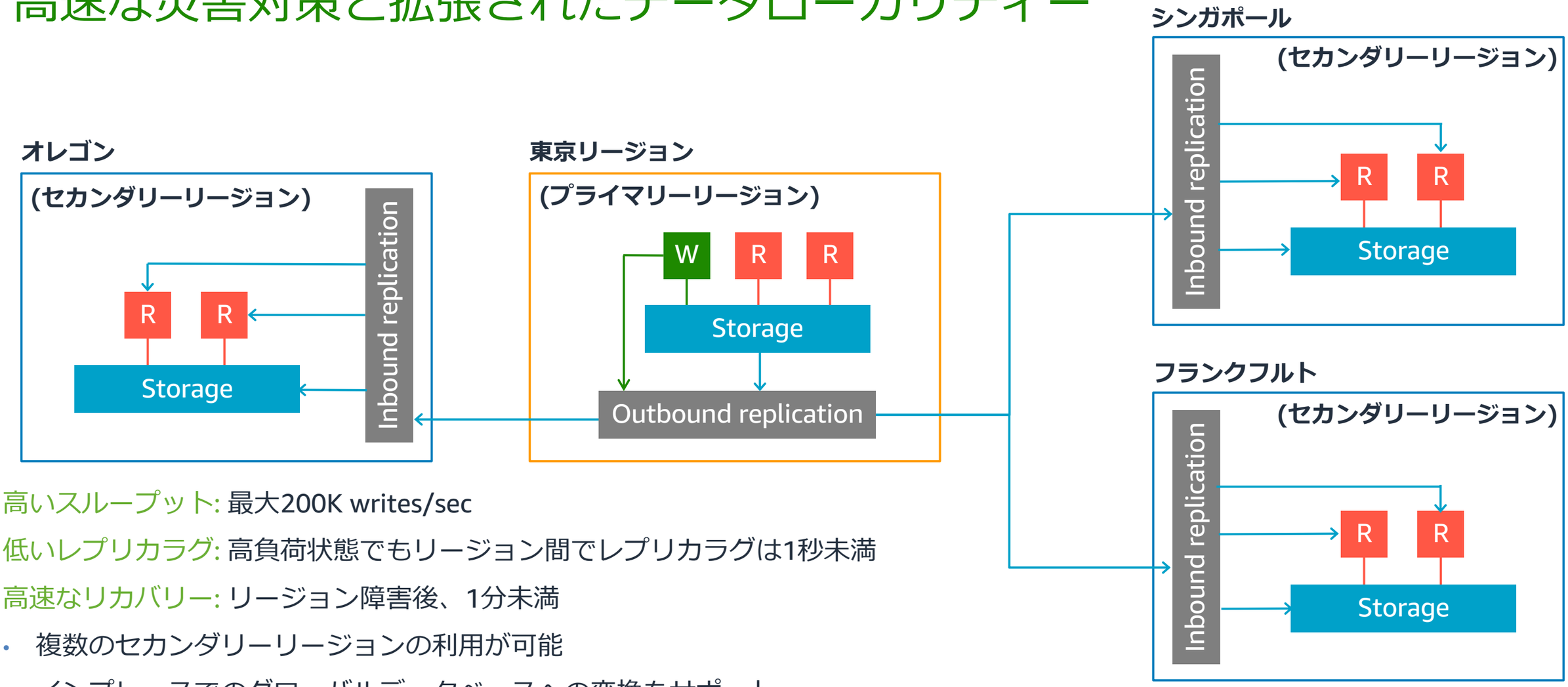
カテゴリ	Aurora MySQL (5.6, 5.7 compatible)	Aurora PostgreSQL (9.6.X, 10.X, 11.X compatible)
可用性	Aurora Multi-Master (5.6 互換のみ) ゼロダウンタイムパッチ (ZDP)	Cluster Cache Management
パフォーマンス	Parallel Query (5.6 互換のみ) 再実装したクエリキャッシュ Asynchronous key prefetch Adaptive Thread Pool Latch-free lock manager	Query Plan Management
データリカバリ	バックトラック	
AWS サービスとの インテグレーション	Amazon S3 import/export Lambda 関数の呼び出し	Amazon S3 import (>= 10.7)

※ オレンジ色のものが、セッション中でご紹介するもの

Aurora グローバルデータベース

高速な災害対策と拡張されたデータローカリティー

MySQL / PostgreSQL



高いスループット: 最大200K writes/sec

低いレプリカラグ: 高負荷状態でもリージョン間でレプリカラグは1秒未満

高速なリカバリー: リージョン障害後、1分未満

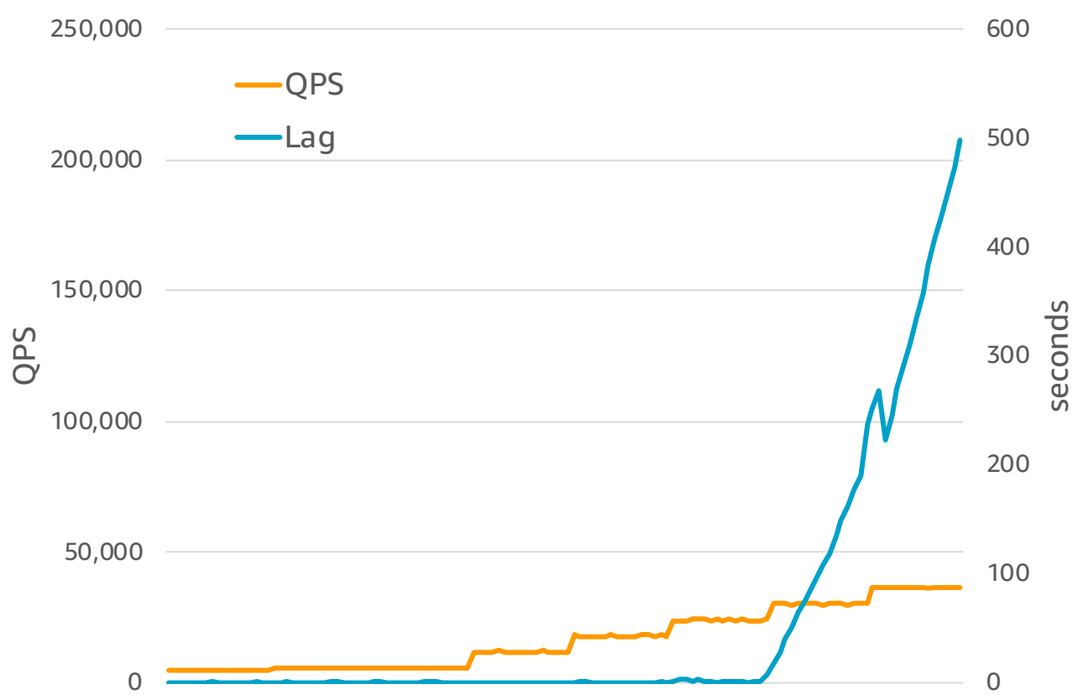
- 複数のセカンダリーリージョンの利用が可能
- インプレースでのグローバルデータベースへの変換をサポート

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/aurora-global-database.html

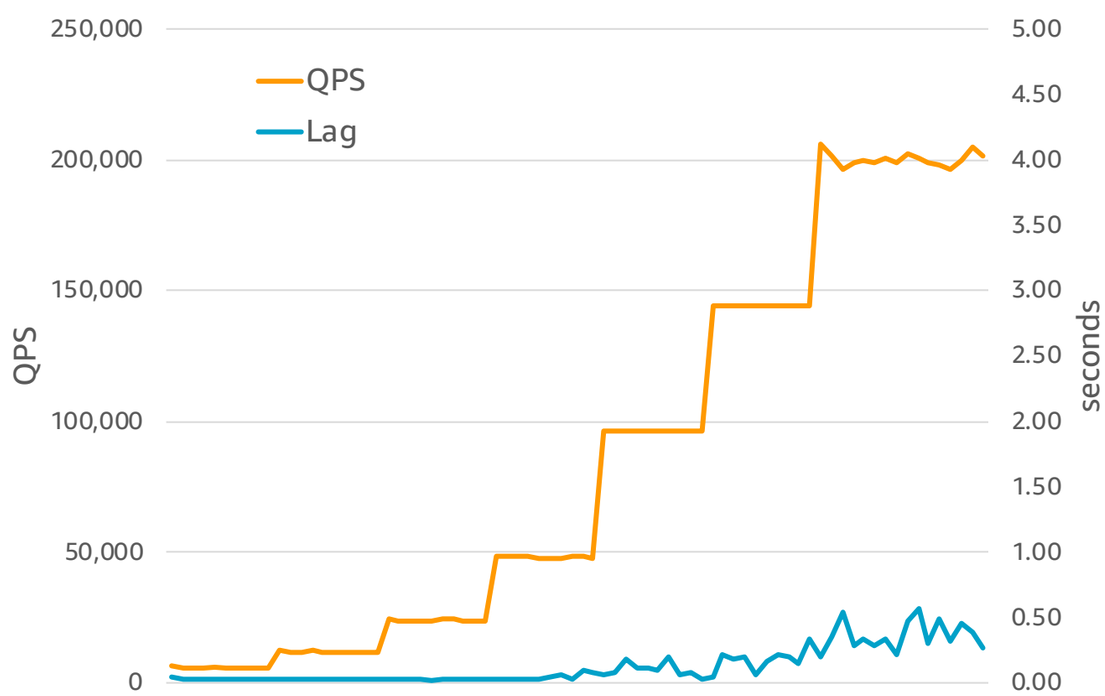
グローバルデータベースのパフォーマンス

MySQL / PostgreSQL

Logical (binlog) vs. Physical (GD) MySQL replication



Logical replication with MTS



Physical replication (GD)

SysBench OLTP (write-only) stepped every 600 seconds on R4.16xlarge

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/aurora-global-database.html

データベースの高速なクローン作成

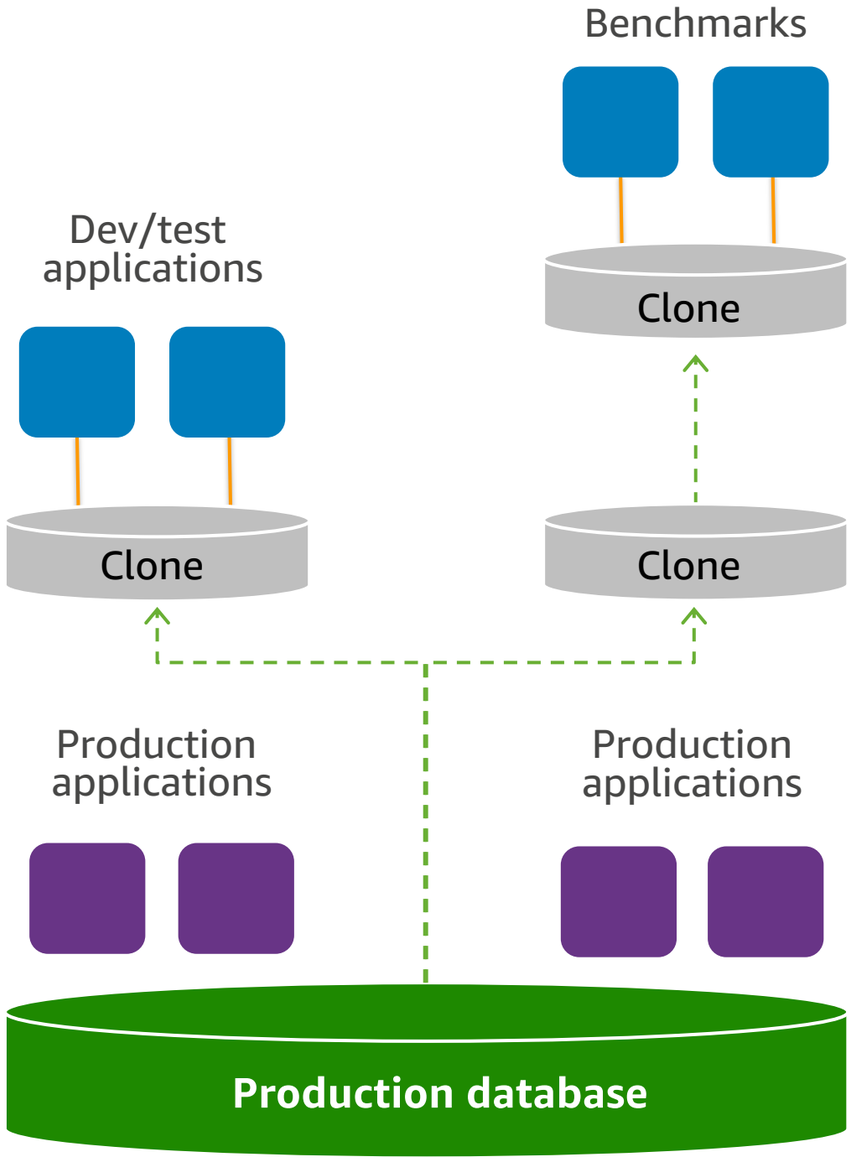
MySQL / PostgreSQL

ストレージコストを増やすことなく データベースのコピーを作成 (Copy on Write)

- データコピーをしないため、数 TB の DB でも クローンの作成はほぼ即座に完了
- クローンボリュームにデータを書き込む際に、 オリジナルデータのコピーが行われる
- アカウント間のクローン共有も可能
- 最大15個のクローンを作成可能

ユースケース

- テスト環境用に実稼働 DB のクローンを作成する
- データベースを再編成する
- 本番環境に影響を与えることなく、分析のために 特定の時刻のスナップショットを保存する

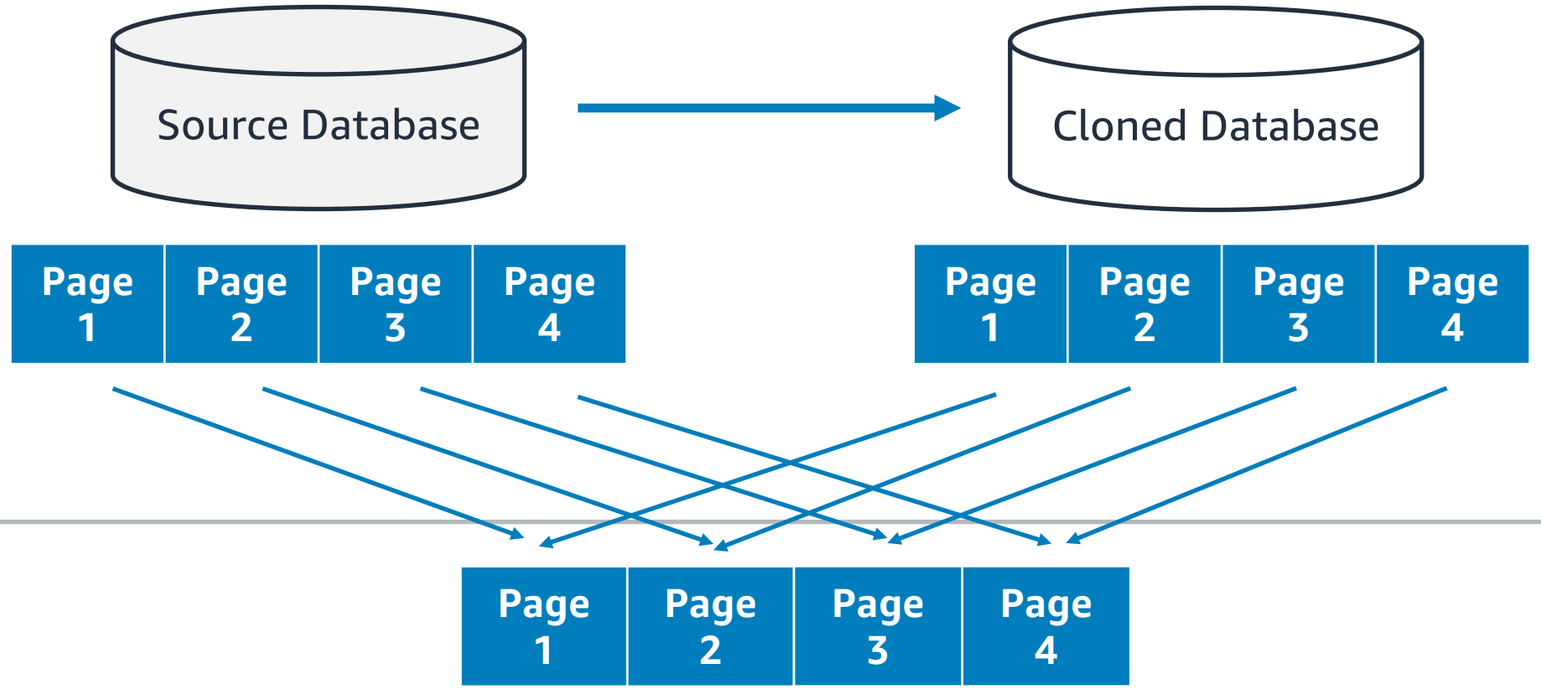


詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Clone.html

クローンのしくみ 1/2

MySQL / PostgreSQL

ストレージコストの重複無しに DB クラスターを作成する
物理的なデータコピーを行わないので、作成は高速



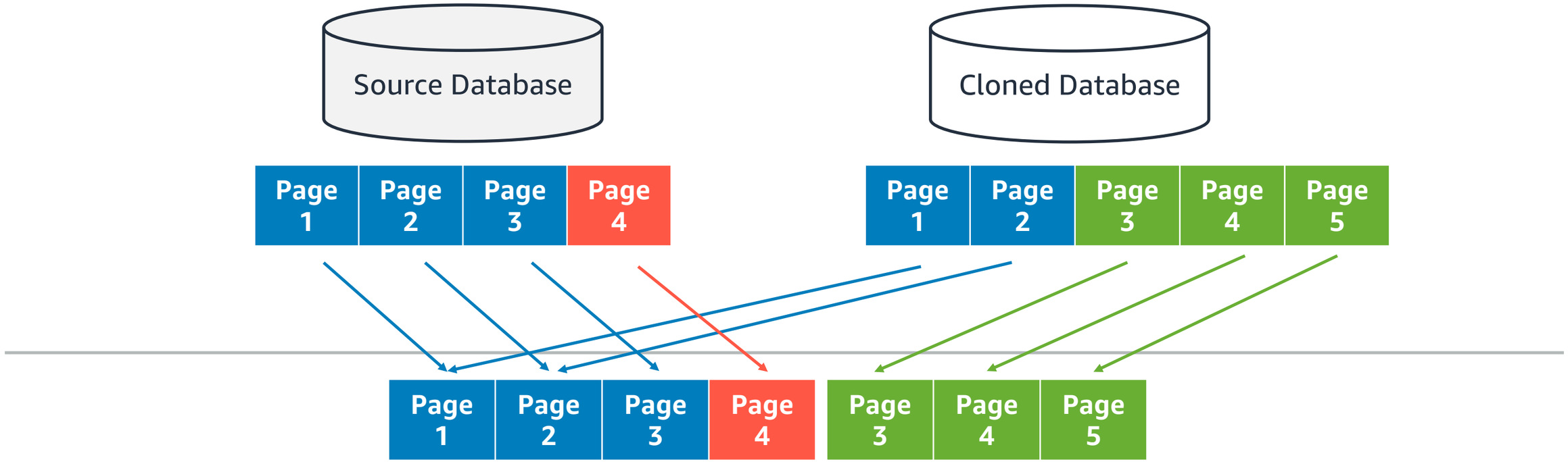
クローンを作成して、まだストレージに変更がない状態
両方のデータベースは、分散共有ストレージ上の同じページを参照する

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Clone.html

クローンのしくみ 2/2

MySQL / PostgreSQL

それぞれのデータベース上の操作は、お互いのパフォーマンスに影響を与えない



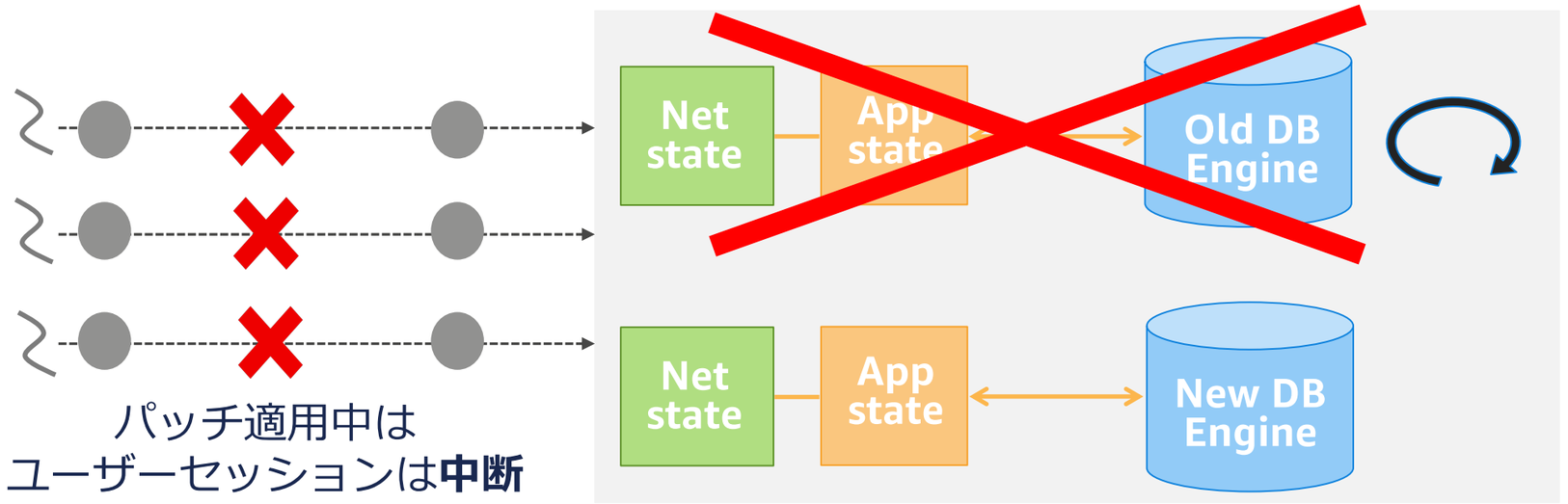
クローンを作成し、ソースとクローンの両方でストレージを変更した状態
両方のデータベースは、 分散共有ストレージ上の共通するページを参照する

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Clone.html

ゼロダウンタイムパッチ (ZDP)

MySQL / PostgreSQL

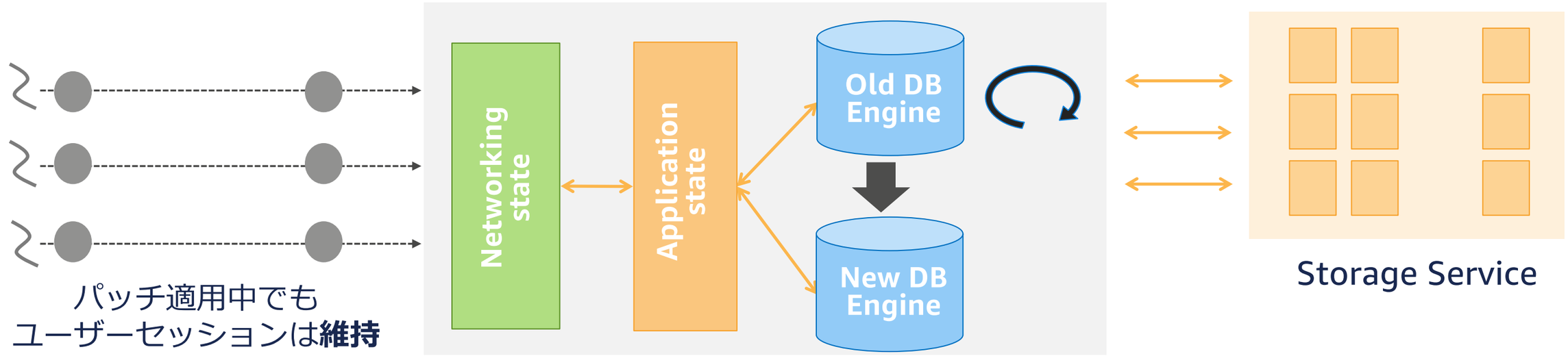
Before ZDP



パッチ適用中は
ユーザーセッションは**中断**

* ゼロダウンタイムパッチはベストエフォート, Writer のみ適用

With ZDP

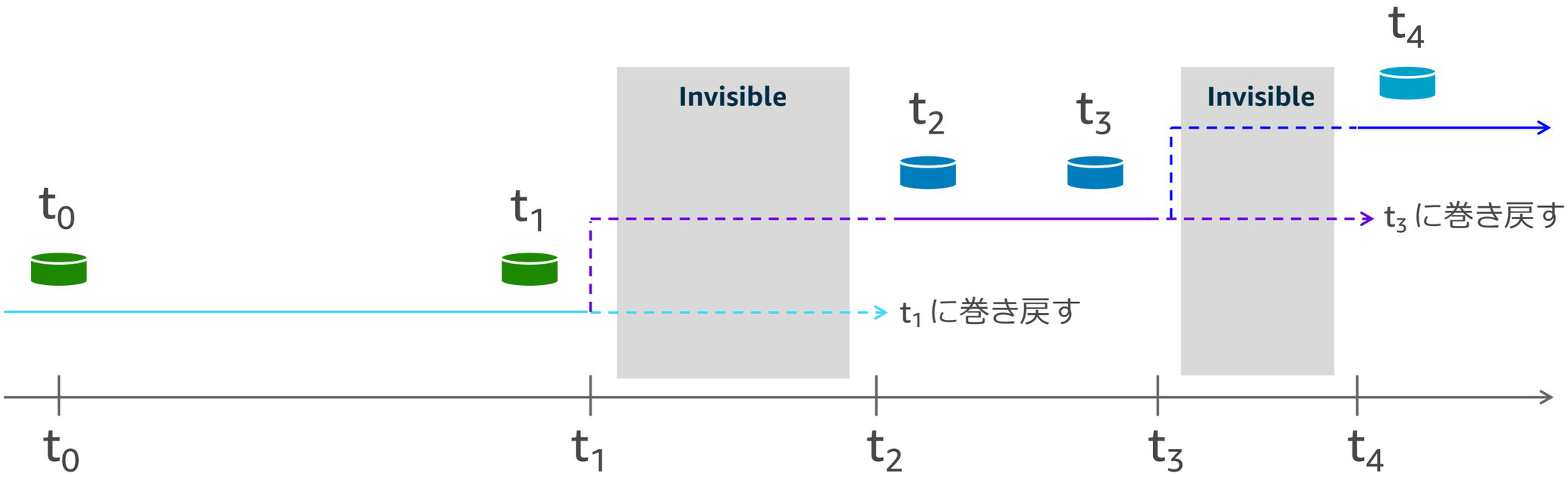


パッチ適用中でも
ユーザーセッションは**維持**

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Updates.html#AuroraMySQL.Updates.ZDP

Aurora バックトラック

MySQL / PostgreSQL



バックトラックは、バックアップからリストアを実施せずに
データベースを高速に特定の時点に戻します

- 意図しない DML または DDL 操作から迅速に復旧
- バックトラックは破壊的操作ではない。複数回バックトラックを実行して、適切なポイントを見つけることが可能
- QA にも利用できる (テスト実行の間に DB を巻き戻す)

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Managing.Backtrack.html

Aurora バックトラック

Target backtrack window

- DB クラスタを Backtrack できる時間
- 最大72時間

Actual backtrack window

- DB クラスタを Backtrack できる実際の時間
- ワークロードにより、Target backtrack window よりも短くなる可能性がある

Aurora は常に一貫性のある時間に Backtrack する

- Backtrack の時間を指定すると、Aurora は自動的に最も近い一貫性のある時間を選択するため、完了した Backtrack は指定した時刻と正確に一致しない可能性がある

100万 change レコードにつき、$0.014 (東京リージョン)

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/AuroraMySQL.Managing.Backtrack.html

Parallel Query

MySQL / PostgreSQL

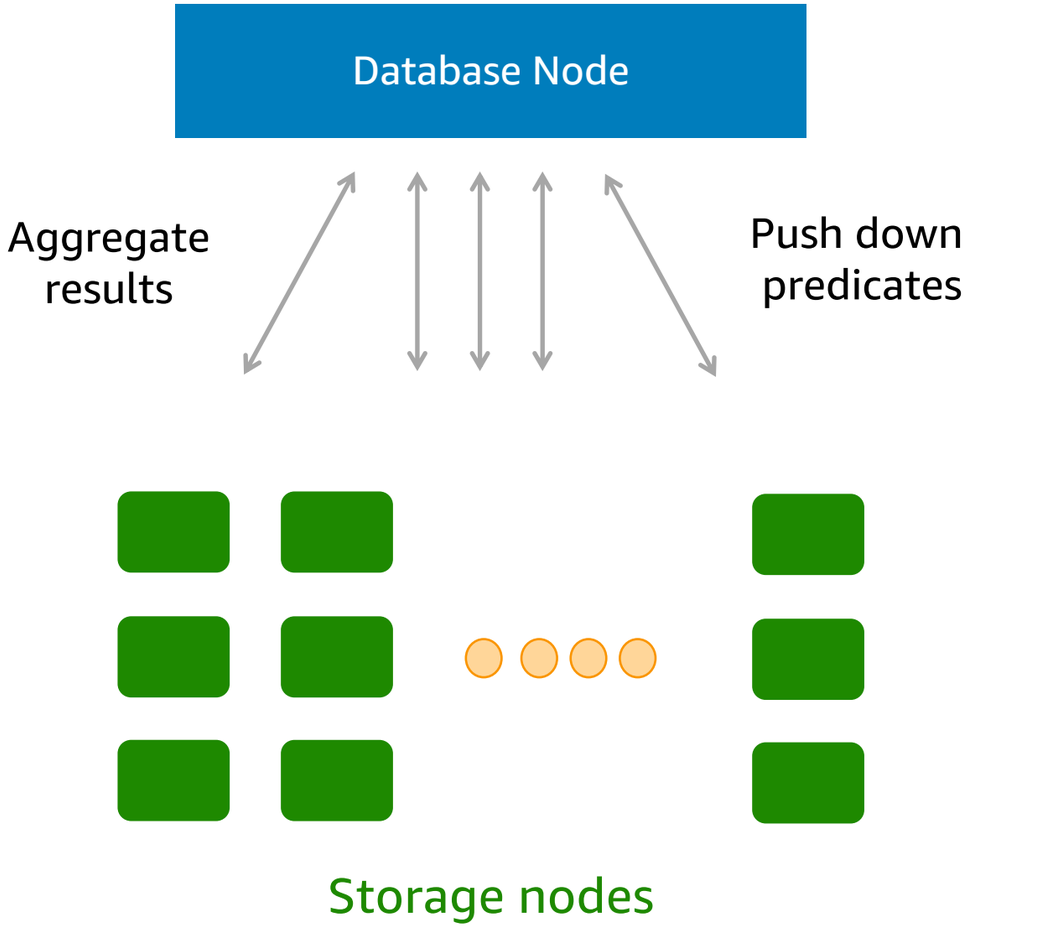
Auroraストレージは数千のCPUを持つ

- クエリ処理をプッシュダウンして並列化
- 処理をデータの近くに移動すると、ネットワークトラフィックとレイテンシが減少する

実現のためには多くのチャレンジがある

- データがレンジパーティションでない場合フルスキャンが必要
- データが未確定の可能性
- 読み取りビューで、最新データを表示できない場合がある
- すべての機能をプッシュダウンできるわけでない

(PostgreSQL は標準的にデータベースエンジン側でサポート)

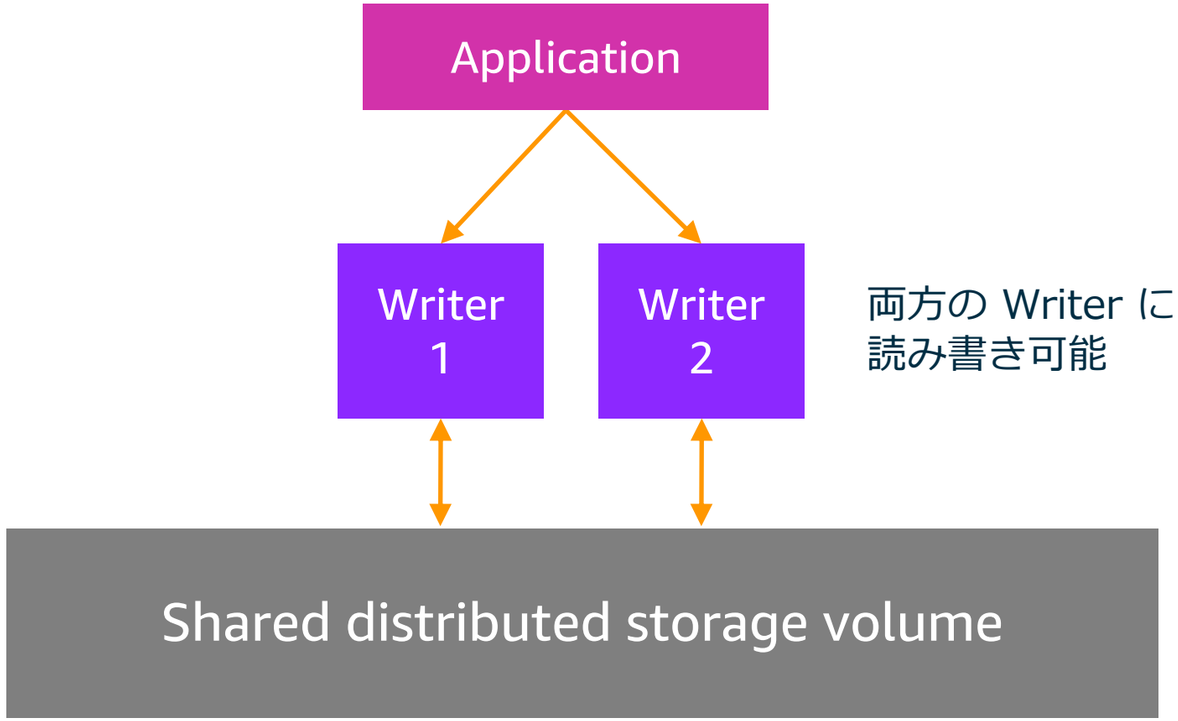


詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/aurora-mysql-parallel-query.html

Aurora Multi-Master による継続的な可用性

MySQL / PostgreSQL

- クラスター内のすべてのノードが、読み取りと書き込み要求の両方に対応できるライターノード
- 共有分散ストレージボリューム
- インスタンスはそれぞれ個別に障害の可能性があり、リカバリーが可能
- 追加費用は発生しない



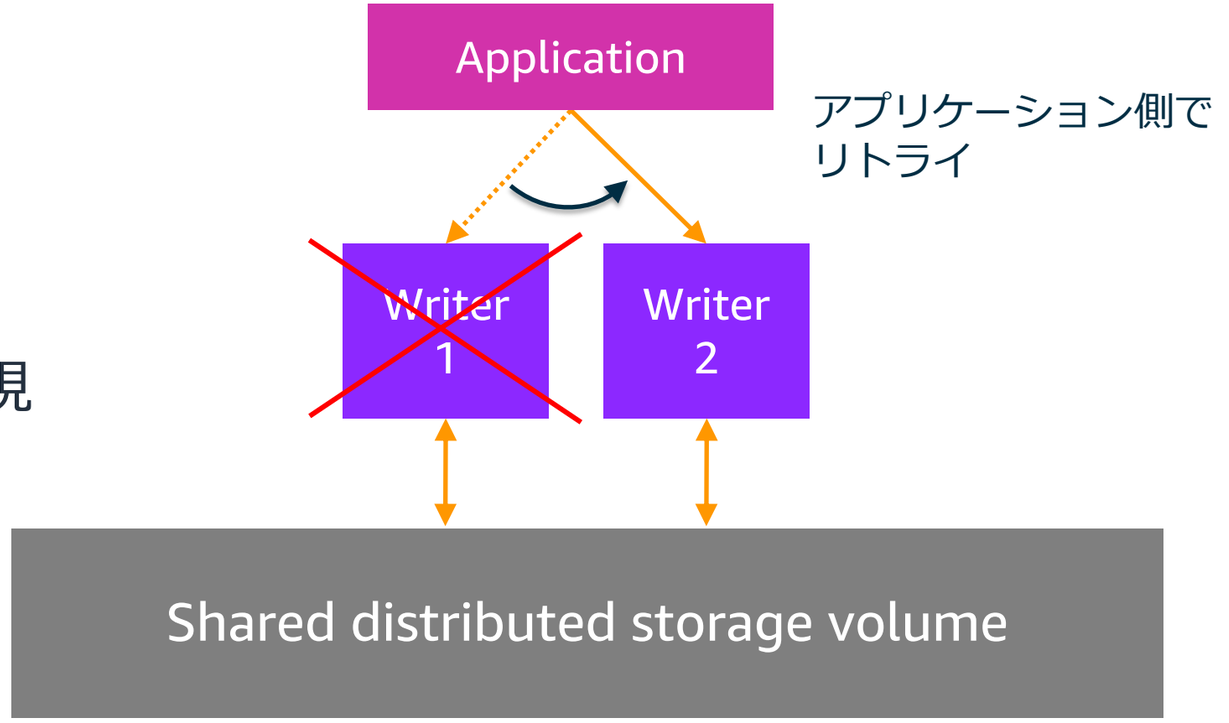
＊ 書き込みのスケールアウトを意図したものではなくあくまでも高可用性が目的

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/aurora-multi-master.html

Aurora Multi-Master による継続的な可用性

MySQL / PostgreSQL

- アプリケーション側にライターノードのヘルスチェックを実装
- 書き込み失敗時、もう片方のライターへリトライすることで継続的な可用性を実現
- 書き込みの競合はページ単位で、トランザクションのコミット時ではなく即座に検出



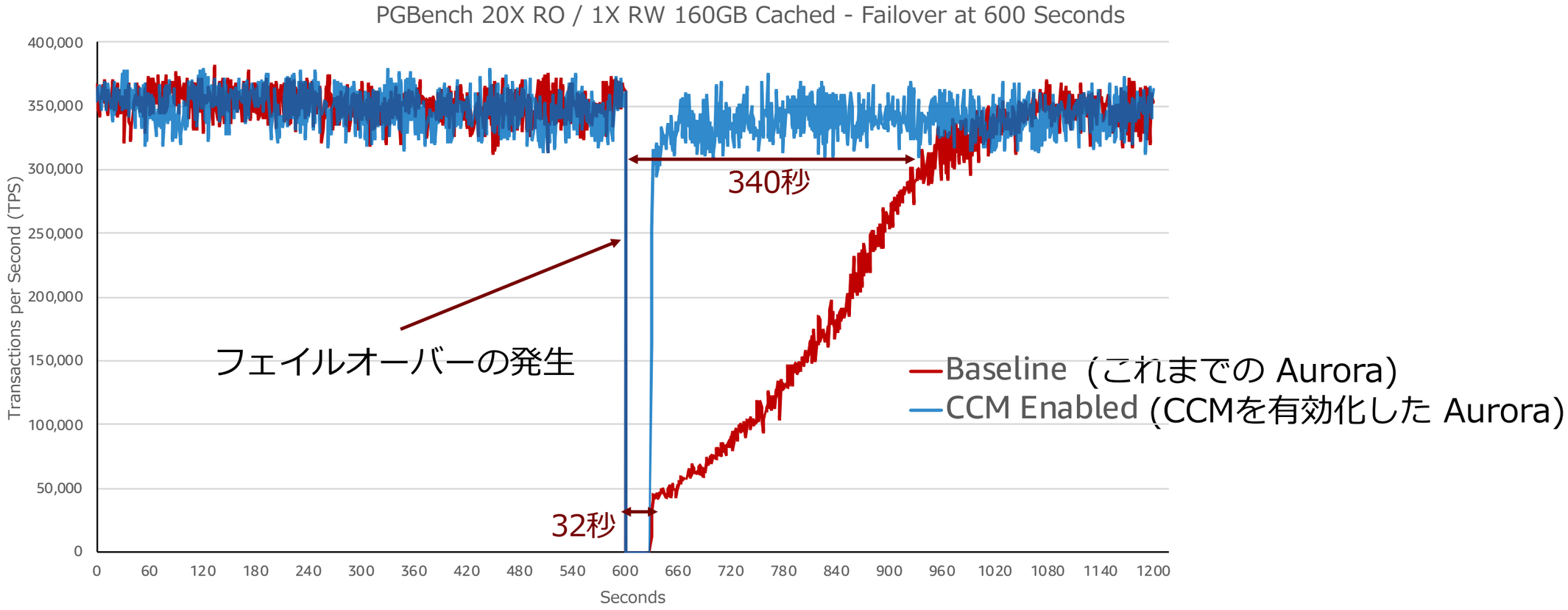
＊ 書き込みのスケールアウトを意図したものではなくあくまでも高可用性が目的

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/aurora-multi-master.html

Cluster Cache Management (CCM)

MySQL / PostgreSQL

Writer とほぼ同期されたウォームキャッシュを持つレプリカへフェイルオーバー

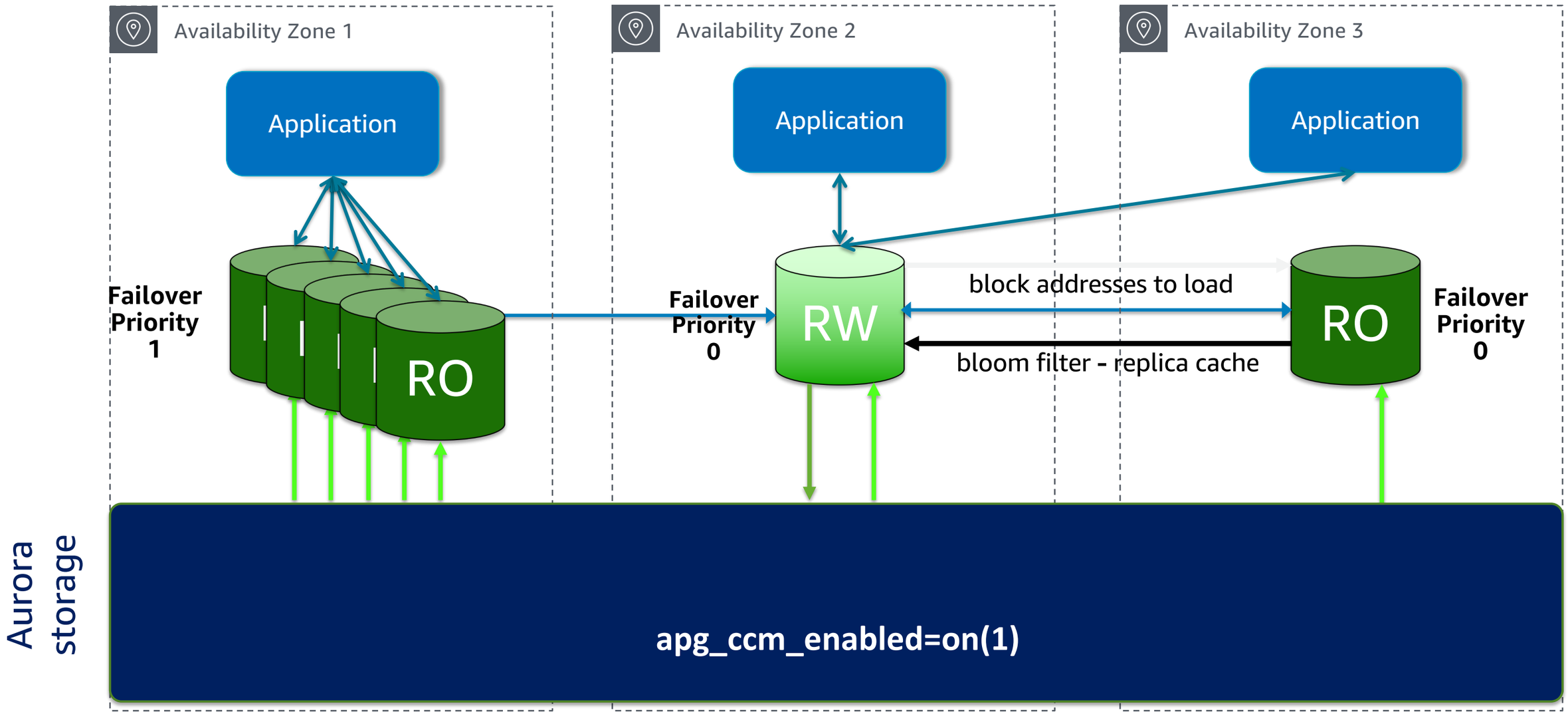


CCM が有効になっていると、データベースはウォームアップされたキャッシュに
フェイルオーバー。フェイルオーバーから32秒後には、90%のパフォーマンスを回復

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.cluster-cache-mgmt.html

Cluster Cache Management (CCM)

MySQL / PostgreSQL



詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.cluster-cache-mgmt.html

Query Plan Management (QPM)

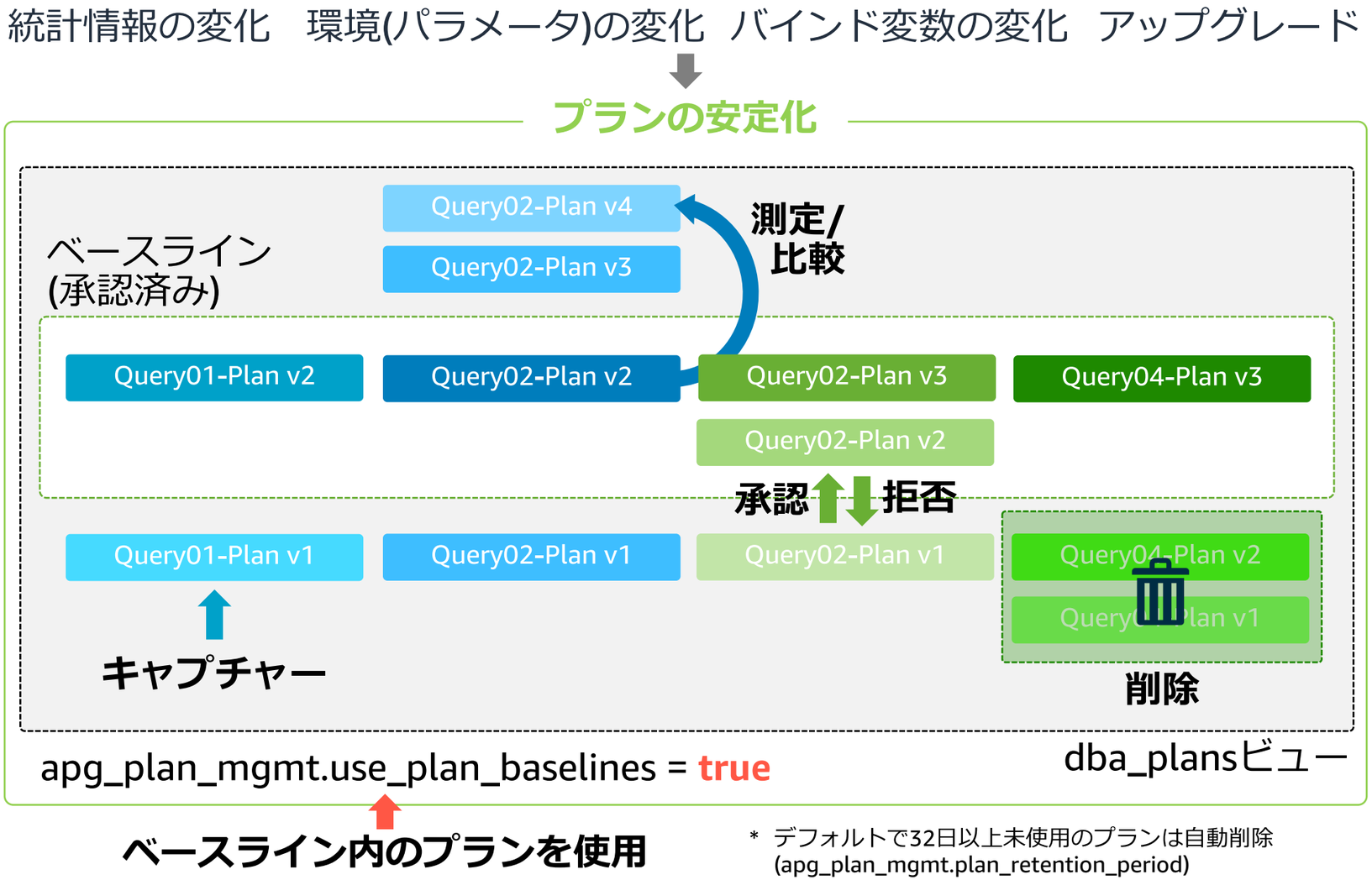
MySQL / PostgreSQL

機能概要

- ✓ 手動/自動でプランのキャプチャー
- ✓ プランの測定/比較
- ✓ プランの承認/拒否
- ✓ ベースライン内のプランを使用
- ✓ pg_hint_planを使ったプランの修正
- ✓ プランの削除
- ✓ プランのエクスポート/インポート

サポートバージョン/制限

- ✓ Aurora PostgreSQL 2.1.0以上 (PostgreSQL 10.5互換)
- ✓ PL/pgSQLは未サポート



詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.Optimize.html

* デフォルトで32日以上未使用のプランは自動削除 (apg_plan_mgmt.plan_retention_period)

* デフォルトで最大1,000個のプランをキャプチャー (apg_plan_mgmt.max_plans)

QPM のユースケース

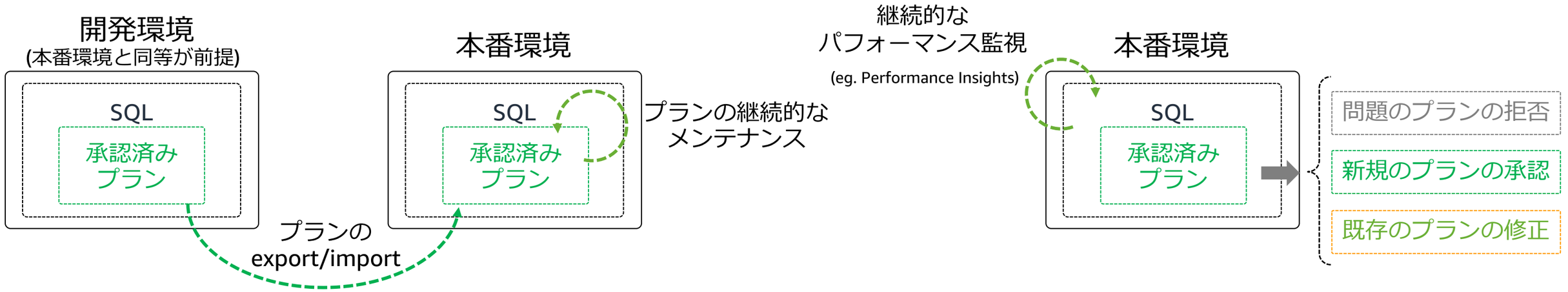
MySQL / PostgreSQL

パフォーマンス低下を防止する事前予防

- 開発環境でパフォーマンスに影響を与えるSQL文を特定し 手動/自動でプランをキャプチャー
- 開発環境から承認済みプランを エクスポート し、本番環境に インポート
- 本番環境では承認された ベースラインのプランを強制
- 新しくキャプチャーされたプランの効率性を分析 し、必要な場合は承認する

パフォーマンス低下を検出した際の事後対応

- アプリケーションのプランを ベースラインで固定 しつつ、新しいプランのキャプチャーを継続
- 実行中のアプリケーションの パフォーマンス低下を監視、分析 (ex. Performance Insights)
- 既存のベースラインのプランを 拒否 し、適切な別の承認済みプランを使用させる
- pg_hint_plan拡張 でプランを 修正 することも可能



詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/AuroraPostgreSQL.Optimize.html

Aurora Serverless

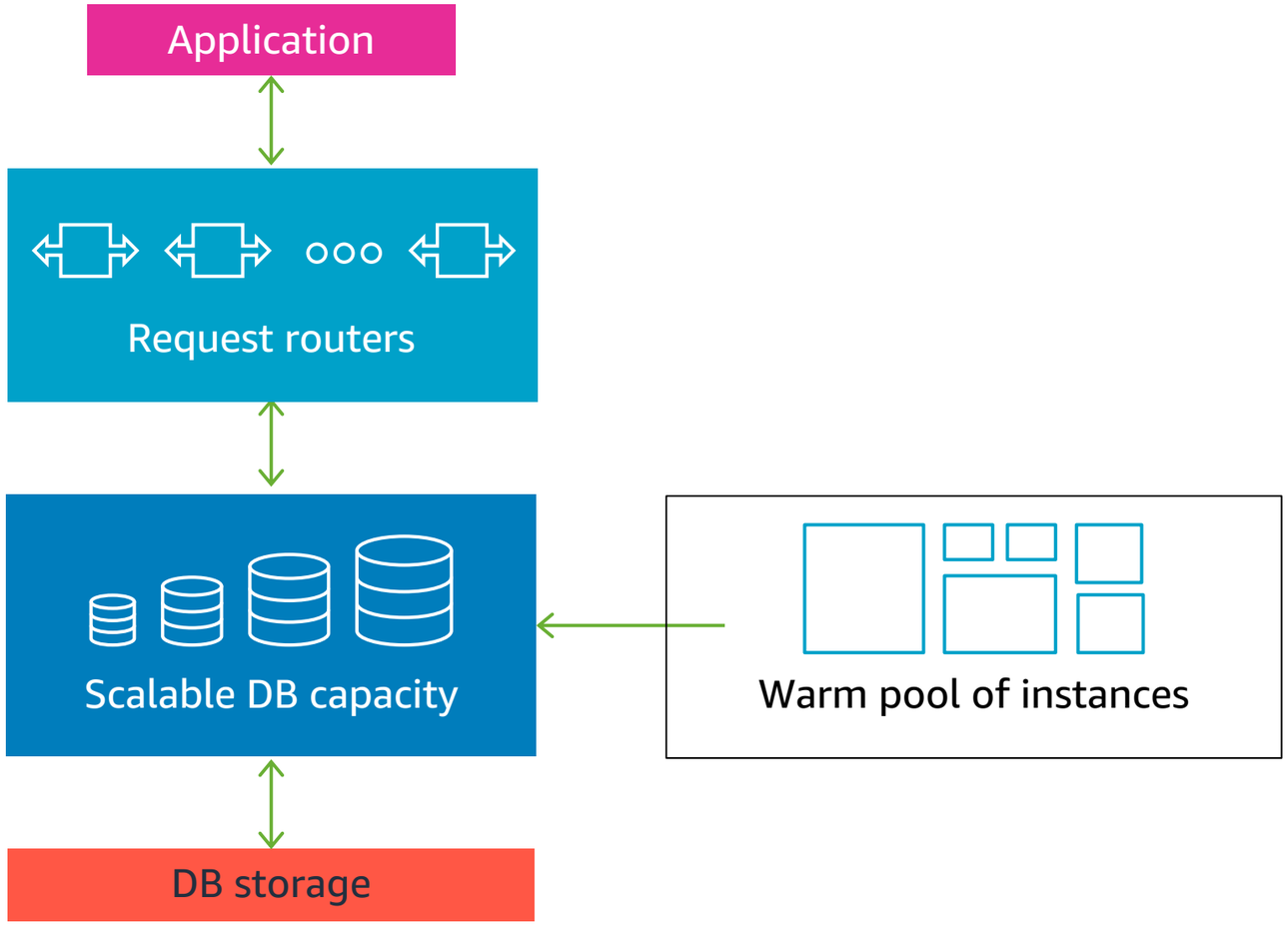
MySQL / PostgreSQL

特徴

- コンピューティングキャパシティーが自動管理される Amazon Aurora
  - オンデマンドで起動
  - 利用状況に応じたにスケーリング
  - 利用されていない場合は自動停止
- スケール時のクライアント接続の中断なし
- 秒課金 (ただし、1分が最低利用料金)

ユースケース

- 不定期利用のアプリケーション
- 開発・テスト用途
- リクエスト予測が難しい場合
  ※ 定常的にリクエストが予測できる場合は通常の Aurora(*) の利用がおすすめ

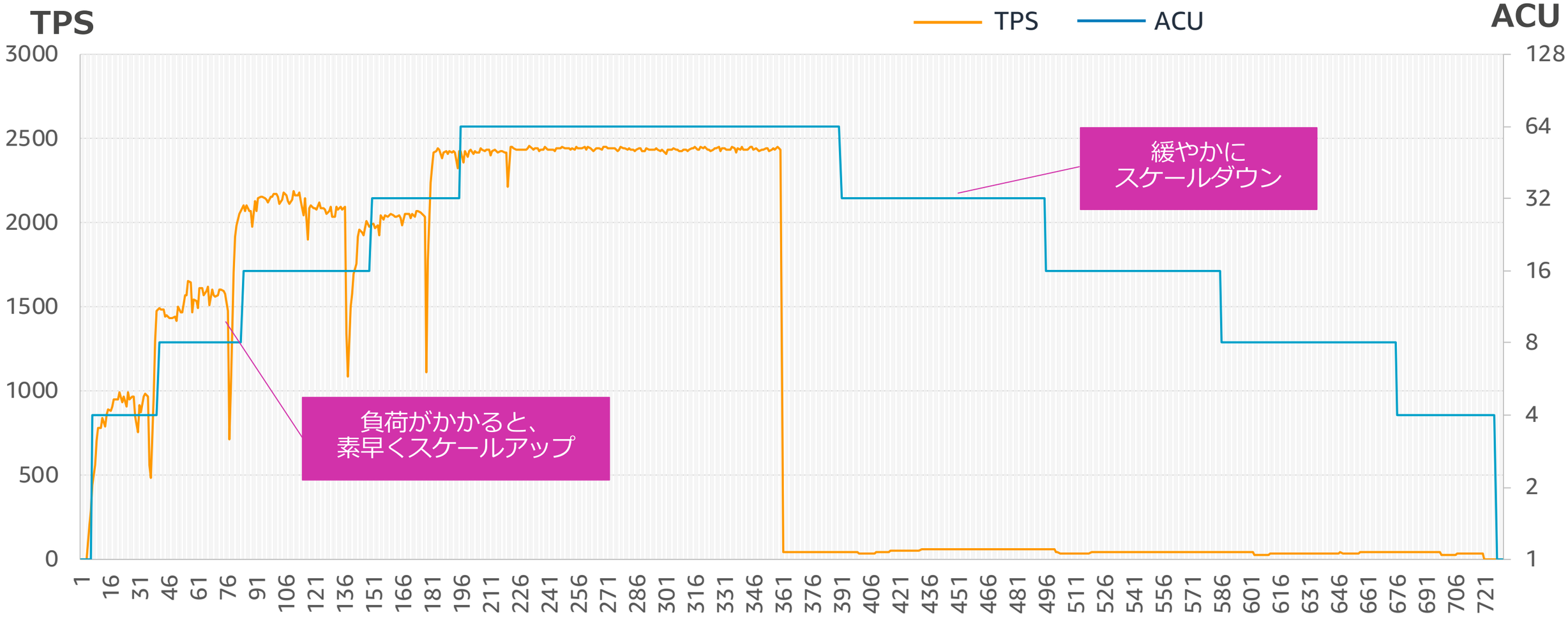


(*) 通常の Aurora (サーバーレスではないAurora) を provisioned DB cluster と呼びます

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/aurora-serverless.html

負荷に応じたスケールアップ・ダウン

MySQL / PostgreSQL

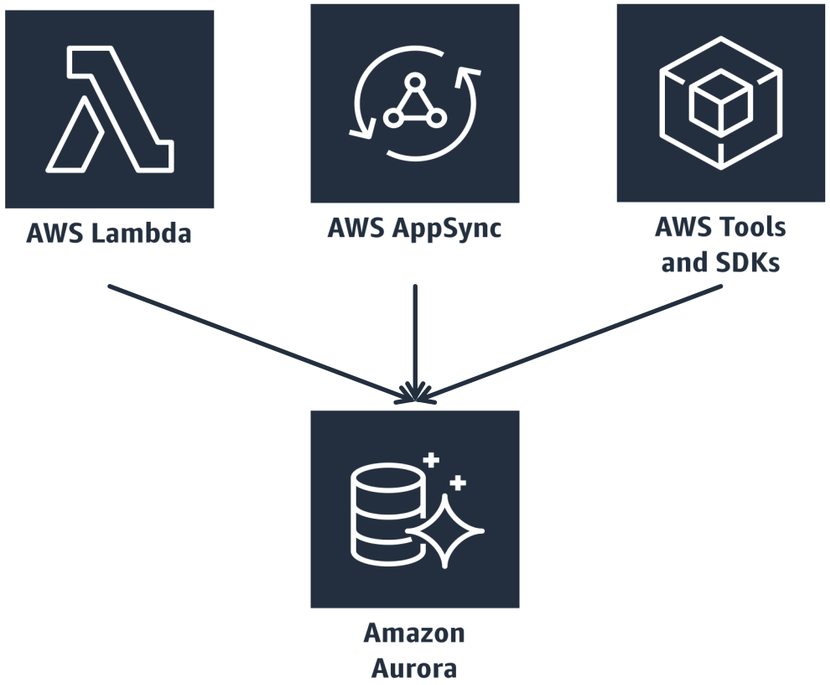


詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/aurora-serverless.html

Aurora Serverless Data API

MySQL / PostgreSQL

- Aurora Serverless へのアクセス方法として、各エンジンのネイティブプロトコルに加え、HTTPS エンドポイントおよび AWS SDK からのアクセスを提供
- AWS Lambda や AWS AppSync から、VPCにアクセスすることなくAuroraを利用可能
- クエリー結果はJSON形式
  - レスポンスは 1,000 行および 1MB が上限
- トランザクションやバッチ実行も専用の API で利用可能

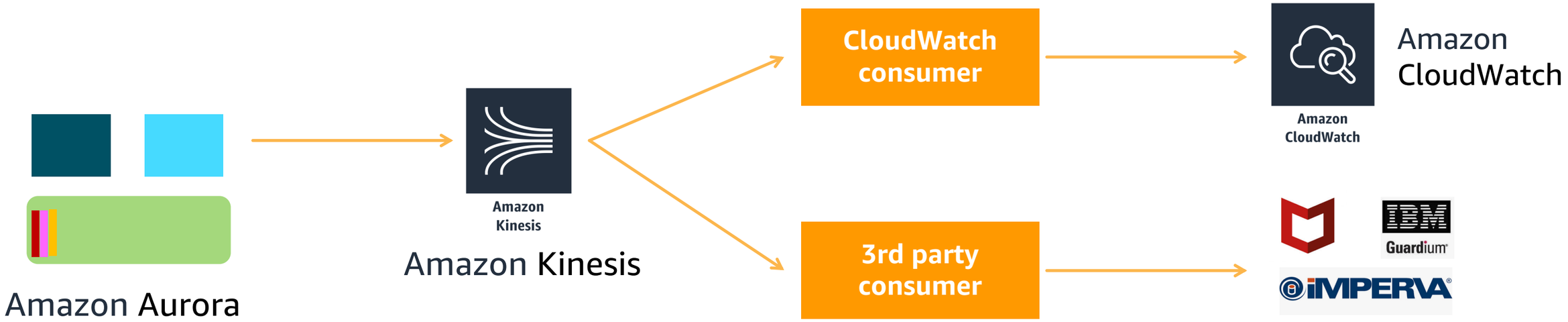



```
aws rds-data execute-sql --db-cluster-or-instance-arn arn:aws:rds:us-east-1:123456789012:cluster:mydbcluster --schema "" \
--database "mydatabase" --aws-secret-store-arn "arn:aws:secretsmanager:us-east-1:123456789012:secret:mysecret" \
--sql-statements "select * from mytable" --region us-east-1 --no-verify-ssl \
--endpoint-url https://rds-data.us-east-1.amazonaws.com --profile myprofile
```

Database Activity Stream (DAS)

MySQL / PostgreSQL

データベースを保護し、コンプライアンス/規制要件を満たすためのモニタリング



- 暗号化済み監査ログを CloudWatch Logs に送信して、DB クラスターのアクティビティを継続的に監視
- アーカイブ用にS3にエクスポート; Amazon Athena を使用したログ分析; Amazon QuickSight でログを可視
- PostgreSQL の場合、以下のパートナー製品が本機能を利用した監査に対応
  - SecureSphere Database Audit and Protection (Imperva)
  - Data Center Security Suite (McAfee)
  - Infosphere Guardium (IBM)

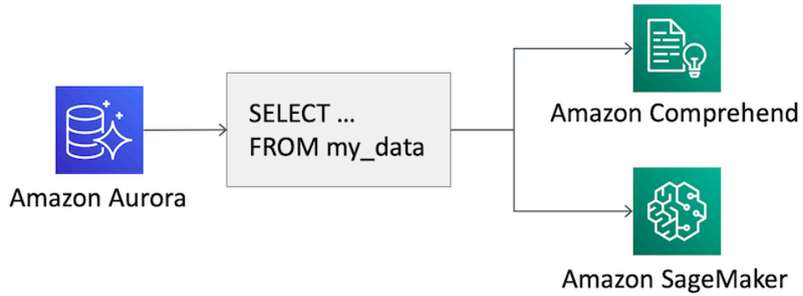
DAS 以外に DB 標準の監査機能も利用可能

詳細：https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/AuroraUserGuide/DBActivityStreams.html

Amazon Aurora ML

MySQL / PostgreSQL

- Amazon Aurora 内から、Amazon SageMaker と Amazon Comprehend を呼び出せるように
  - Amazon Aurora 内に保存されたデータに対して、データを移動するパイプラインを作成することなく、Amazon SageMaker や Amazon Comprehend にデータを投入して、結果を取得可能
  - Stored Function として実行するため、通常の SQL を利用可能
- ユースケース
  - 課金ログから不正トランザクション検出
  - Blog などについたコメントに対してセンチメント分析
  - ユーザ情報などからポテンシャルカスタマーの抽出



詳細：https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-ml.html

Amazon Aurora ML : Amazon Comprehend との連携例

MySQL / PostgreSQL

```
CREATE TABLE IF NOT EXISTS comments (
       comment_id INT AUTO_INCREMENT PRIMARY KEY,
       comment_text VARCHAR(255) NOT NULL
);

INSERT INTO comments (comment_text)
VALUES ("This is very useful, thank you for writing it
INSERT INTO comments (comment_text)
VALUES ("Awesome, I was waiting for this feature.");
INSERT INTO comments (comment_text)
VALUES ("An interesting write up, please add more details.");
INSERT INTO comments (comment_text)
VALUES ("I don't like how this was implemented.");
```

```
SELECT comment_text,
       aws_comprehend_detect_sentiment(comment_text, 'en') AS sentiment,
       aws_comprehend_detect_sentiment_confidence(comment_text, 'en') AS confidence
  FROM comments;
```

```
+-----------------------------------------------+-----------+--------------------+
| comment_text                                  | sentiment | confidence         |
+-----------------------------------------------+-----------+--------------------+
| This is very useful, thank you for writing it! | POSITIVE  | 0.9996347427368164 |
| Awesome, I was waiting for this feature.      | POSITIVE  | 0.9977971315383911 |
| An interesting write up, please add more details. | NEUTRAL | 0.5156506896018982 |
| I don't like how this was implemented.        | NEGATIVE  | 0.9982835054397583 |
+-----------------------------------------------+-----------+--------------------+
```

詳細：https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-ml.html

Amazon Aurora ML : Amazon Sagemaker との連携例

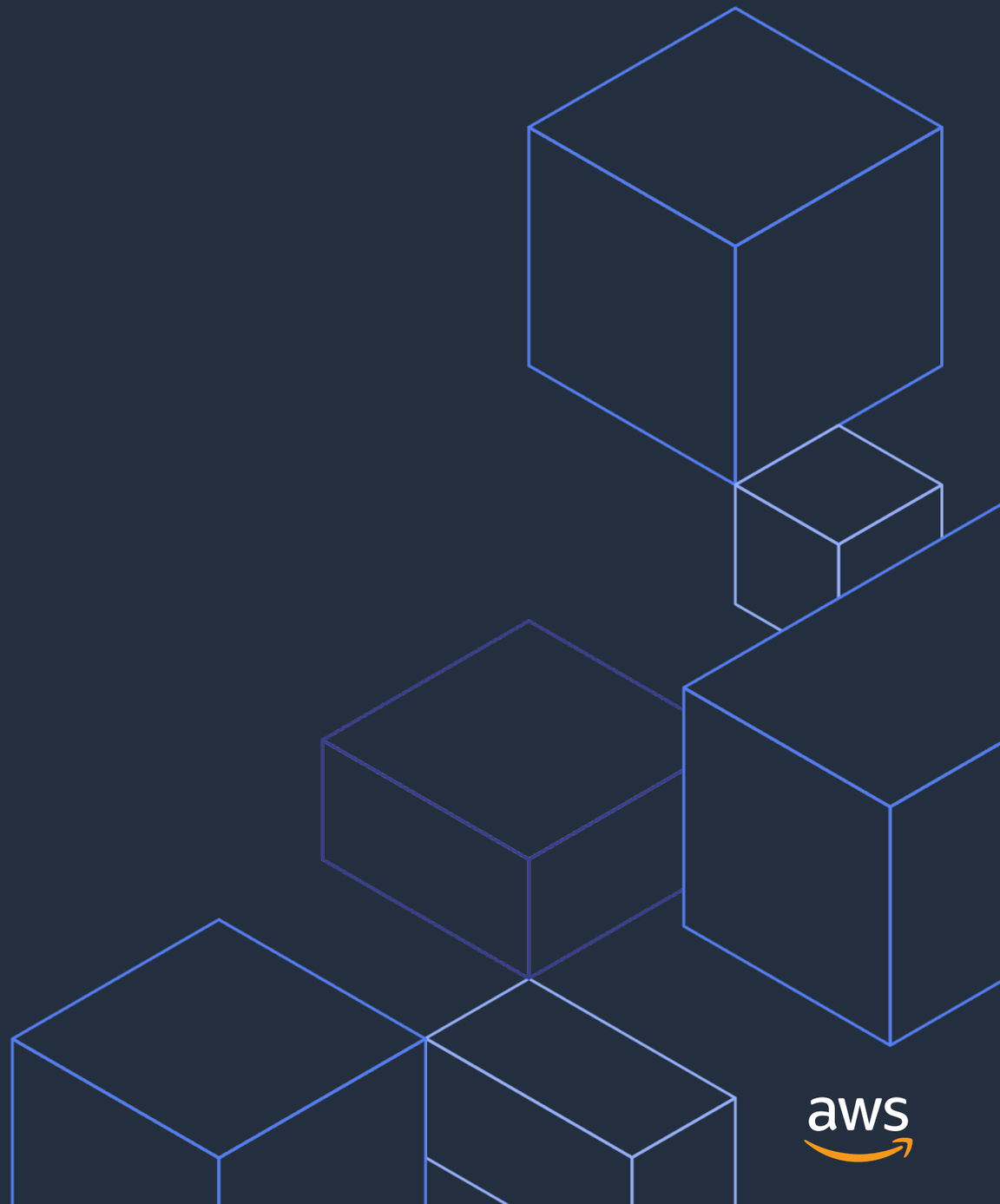
MySQL / PostgreSQL

```
CREATE FUNCTION will_churn (
        state varchar(2048), acc_length bigint(20),
        area_code bigint(20), int_plan varchar(2048),
        vmail_plan varchar(2048), vmail_msg bigint(20),
        day_mins double, day_calls bigint(20),
        eve_mins double, eve_calls bigint(20),
        night_mins double, night_calls bigint(20),
        int_mins double, int_calls bigint(20),
        cust_service_calls bigint(20))
RETURNS varchar(2048) CHARSET latin1
        alias aws_sagemaker_invoke_endpoint
        endpoint name 'estimate_customer_churn_endpoint_version_123';
```

```
CREATE TABLE customers_churn AS
SELECT *, will_churn(state, acc_length, area_code, int_plan,
        vmail_plan, vmail_msg, day_mins, day_calls,
        eve_mins, eve_calls, night_mins, night_calls,
        int_mins, int_calls, cust_service_calls) will_churn
  FROM customers;
```

詳細：https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-ml.html

まとめ



まとめ

- Amazon Aurora はマネージドサービスとして、開発者・運用者にとって重労働となっていた作業を軽減するような、便利な機能が存在する

- パフォーマンスだけでなく、運用面や、システムのアーキテクチャを簡単にするための機能にも注目してみましょう

Thank you!