

AWS 秋の COST OPTIMIZATION 祭り 2025

Aurora のコスト構造を理解して最適化 ～技術的アプローチと実践～

西原 陽介

Amazon Web Services Japan G.K.
Technical Account Manager



自己紹介

- AWS 入社前は日系 SIer に在籍
- 現在、AWS Enterprise Support にて、金融業界のお客様を中心に
ご支援しています
- 好きな AWS サービス
 - Amazon Aurora , Amazon Timestream for InfluxDB



目次

- Amazon Aurora のコスト構造
- Amazon Aurora インスタンス最適化
- Amazon Aurora I/O 最適化
- Amazon Aurora Serverless V2

Amazon Aurora のコスト構造

Amazon Aurora のコスト構造

Aurora Global Database

Data Transfer

Backup

Extended Support

Backtrack (Aurora MySQL)

Snapshot

Data API

Export to S3

ユースケースに
依存するコスト

インスタンス
(Provisioned/Serverless)

I/O
(I/O 最適化/スタンダード)

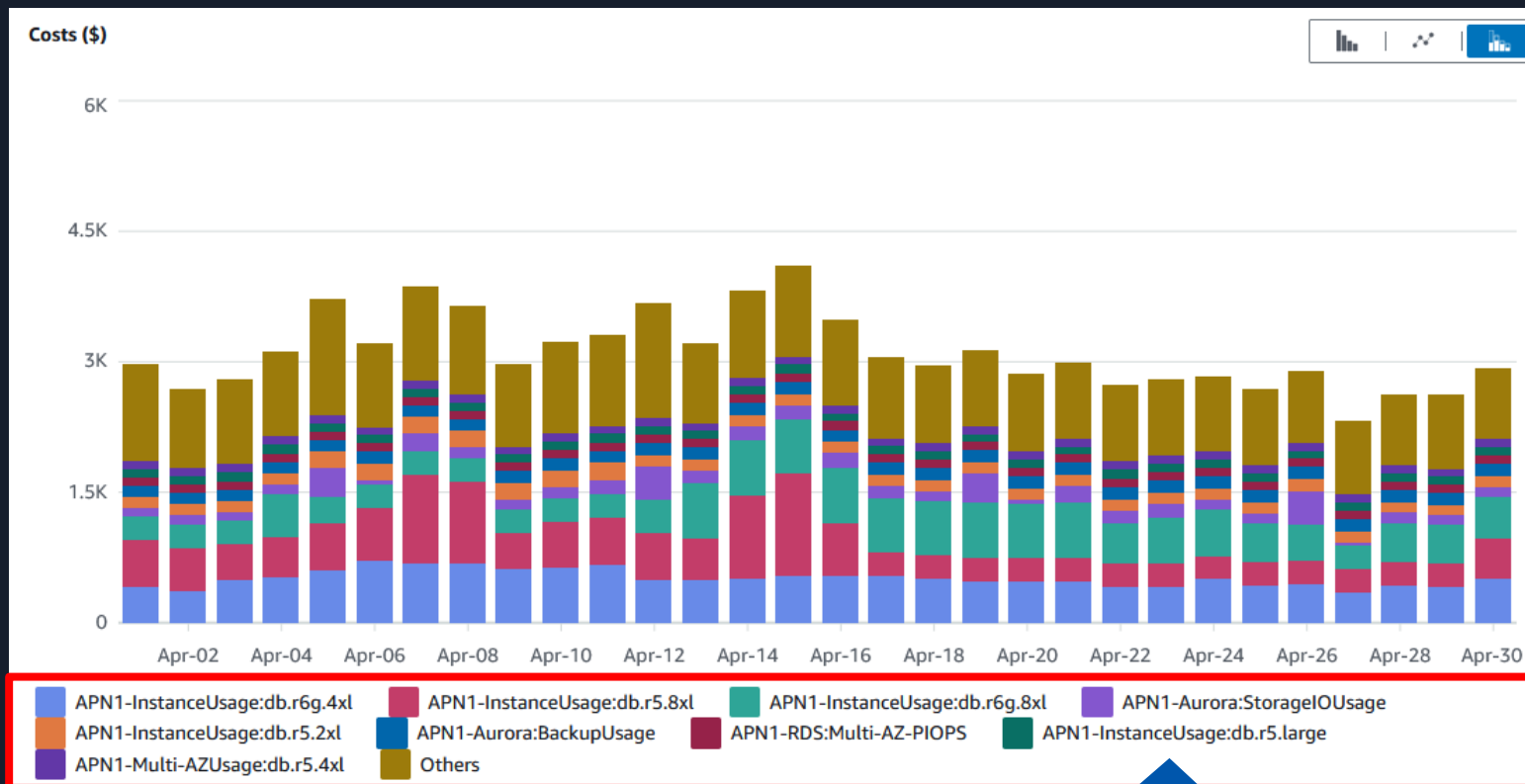
ストレージ

基本的なコスト

今回、説明する箇所

Amazon Aurora のコストの詳細確認

- Cost Explorer の「使用タイプ」を表示する事で、RDS/Aurora の利用料の詳細が確認できます



Aurora の費用の詳細を
確認する事が可能

使用タイプを選択

▼ グループ化の条件

ディメンション クリア

使用タイプ ▼

▼ フィルター 情報

適用フィルター (1) すべてをクリア | ⚙

サービス クリア

サービスを含む (1) ▼

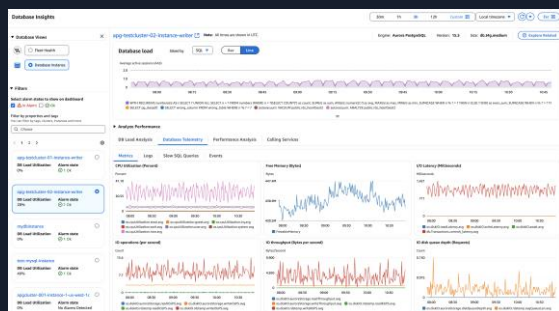
Relational Database Service (RDS) ✕

Amazon Aurora インスタンス最適化 (provisioned)

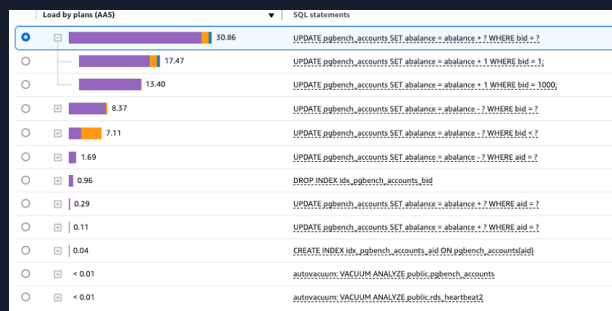
1. インスタンスのダウンサイジング

適切なサイズに変更する事でコスト削減を図る

- CloudWatch Database Insights でリソース状況に余裕のあるインスタンスを確認



CPU 使用率や I/O wait



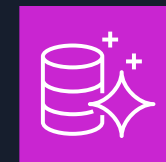
SQL 統計

- Database Insights (standard) は 7 日分のパフォーマンスデータを保持
- より長期間のパフォーマンスデータの履歴を利用する為には Advanced mode にする必要有

<https://aws.amazon.com/cloudwatch/pricing/>



- インスタンスタイプを小さいサイズに変更する



db.r6g.4xlarge
\$ 2.506/h



db.r6g.2xlarge
\$ USD 1.253/h

- 性能影響がない事を事前に確認した上で変更する
- SQL チューニングの余地が無いか確認**頂く事も推奨

<https://aws.amazon.com/jp/rds/aurora/pricing/>

精益求精 (せいえききゅうせい)

すでに優れているものでも、さらなる改善を追求するという考え方

性能改善出来ないか SQL レベルで確認するようにしましょう

Load by plans (AAS)		SQL statements
☒	30.86	UPDATE pgbench_accounts SET abalance = abalance + ? WHERE bid = ?
☐	17.47	UPDATE pgbench_accounts SET abalance = abalance + 1 WHERE bid = 1;
☐	13.40	UPDATE pgbench_accounts SET abalance = abalance + 1 WHERE bid = 1000;
☐	8.37	UPDATE pgbench_accounts SET abalance = abalance - ? WHERE bid = ?
☐	7.11	UPDATE pgbench_accounts SET abalance = abalance - ? WHERE bid < ?
☐	1.69	UPDATE pgbench_accounts SET abalance = abalance - ? WHERE aid = ?
☐	0.96	DROP INDEX idx_pgbench_accounts_bid
☐	0.29	UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?
☐	0.11	UPDATE pgbench_accounts SET abalance = abalance + ? WHERE aid = ?
☐	0.04	CREATE INDEX idx_pgbench_accounts_aid ON pgbench_accounts(aid)
☐	< 0.01	autovacuum: VACUUM ANALYZE public.pgbench_accounts
☐	< 0.01	autovacuum: VACUUM ANALYZE public.rds_heartbeat2

参考: Compute Optimizer – Aurora インスタンスの推奨事項

- Compute Optimizer コンソール画面で有効化する事で利用可能に
- Aurora インスタンスの推奨事項を提示
- Performance Insights を有効とすると、過剰なプロビジョニングとなっているインスタンスと推奨するインスタンスタイプを提示してくれる

パフォーマンスとコスト削減の機会
グループのアイドル状態のステータスと適切なサイズ設定のオプションの概要。

アイドル状態のオプション [情報](#)

アイドル状態のステータス	推定コスト削減額 情報	推奨アクション
☐ アイドル	\$ 259.88 /mo	Stop this DB instance for up to 7 days or create a DB snapshot and delete the instance

適切なサイズ設定のオプション [情報](#)

CPU アーキテクチャの詳細設定: 現在 [設定](#)

オプション	インスタンスタイプ	推定月間節約額 (割引後) 情報	推定月間節約額 (オンデマンド) 情報	節約の機会 (%) 情報	料金 (割引後) 情報
☐ 現在	db.m5.xlarge	-	-	-	\$0.3560 時間あたり
☒ オプション 1	db.t3.xlarge	\$45.3120	\$45.3120	18.54%	\$0.2900 時間あたり

推定節約額

https://docs.aws.amazon.com/ja_jp/compute-optimizer/latest/ug/what-is-compute-optimizer.html



2. インスタンスタイプの変更

世代やプロセッサファミリーを見直す事によりコストパフォーマンス向上を図る

- 最新世代へ移行



db.r5.4xlarge
\$ 2.80/h



db.r7i.4xlarge
\$ 2.80/h

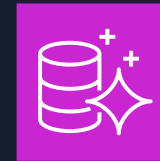
金額は同一だが、
スペックは改善



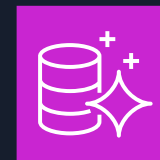
db.r7i.2xlarge
\$ 1.40/h

インスタンスサイズ
縮小を検討

- CPU の変更



db.r6i.4xlarge
\$ 2.80/h



db.r6g.4xlarge
\$ 2.506/h
(△10%)

Graviton CPU が Intel
CPU よりもコア当たり
の単価は低い

3. Reserved Instance (RI) の購入

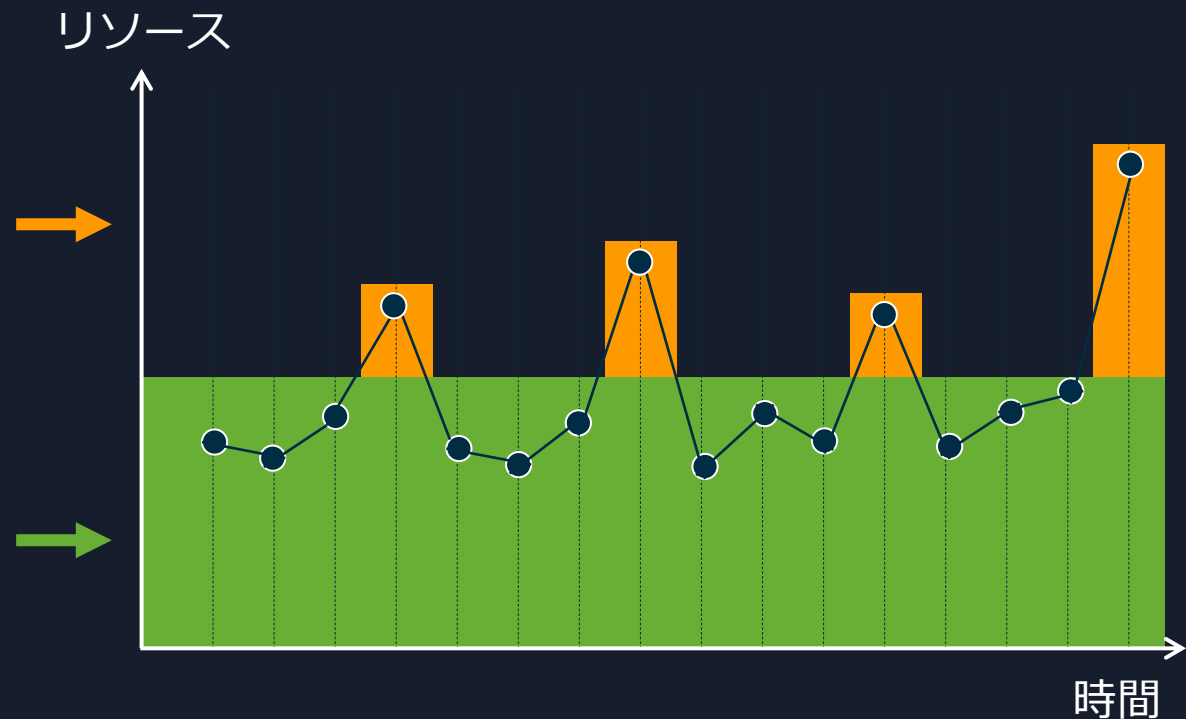
1 年 or 3 年の契約コミットメントにより、オンデマンド料金と比較して最大 69% のコストを削減可能

オンデマンド
インスタンス

初期費用なし、
コミットなし、
従量課金

リザーブド
インスタンス

長期(1年or3年)の
利用コミットに
よる割引の適用
(最大69%割引)



https://docs.aws.amazon.com/ja_jp/AmazonRDS/latest/UserGuide/USER_WorkingWithReservedDBInstances.html

<https://aws.amazon.com/jp/rds/aurora/pricing/>

4.利用しないクラスタの停止

- 開発環境やテスト環境を週末夜間等の利用しない時間帯に停止することでコストを抑える
 - 停止中はストレージ、バックアップのみ課金
 - Instance Scheduler on AWS による自動起動/停止の仕組みも提供
 - <https://docs.aws.amazon.com/solutions/latest/instance-scheduler-on-aws/solution-overview.html>
- 使用頻度が少ない場合、バックアップを取得してクラスタ削除
- Aurora Global Database のヘッドレス構成を検討

<注意点>

- クラスタを停止しても、停止後 7 日間で自動で起動する
- Aurora Global Database の一部であるクラスタの停止は不可
- クロスリージョンリードレプリカを持つクラスタの停止は不可

クラスタ停止時の設定イメージ

DB クラスタを一時的に停止

この database-1 DB クラスタは最大 7 日間停止しようとしています。DB クラスタはいつでも手動で再起動できます。[詳細はこちら](#)

了承する

☒ 私は、DB インスタンスを停止しても、請求が一時停止されるのはインスタンス時間だけであることを承知します。DB インスタンスは May 13, 2025, 09:56 (UTC+09:00) に自動的に再起動します。

⚠

DB クラスタが停止している間、プロビジョンド IOPS を含め、プロビジョンドストレージに対して請求されます。また、バックアップストレージの料金も請求されます。これには、保持期間内の手動スナップショットと自動バックアップが含まれます。詳細については、「[請求に関するよくある質問](#)」を参照してください。

ℹ

Lambda 関数を使用すると、メンテナンス期間の開始時と終了時に DB クラスタを自動的に起動および停止できます。[詳細はこちら](#)。

キャンセル

一時的に停止

aws

© 2025, Amazon Web Services, Inc. or its affiliates. All rights reserved. Amazon Confidential and Trademark.

13

3.RI 購入 vs 4.利用しないクラスタ停止

3.RI 購入

Pros :

- インスタンスにアクセス不要で手軽に割引が得られる

Cons :

- 長期の利用コミットが必要となり、購入キャンセルが出来ない

1年：全額前払い
\$ 1,840 /年 (※)

4.利用しないクラスタの停止

Pros :

- 業務要件に応じて停止時間を柔軟に設定できる
- 長期間のインスタンス利用のコミットが不要

Cons :

- 起動・停止処理の開発は必要
- キャパシティエラーの可能性がゼロではない

AM 8 時～ PM 11 時の間のみ起動
÷ **\$ 1,820 / 年 (※)**

※ Aurora PostgreSQL、東京リージョン、r8g.large の場合

Amazon Aurora インスタンス最適化 (provisioned)

事前確認

Cost Explorer
Aurora 費用確認

Compute Optimizer

CloudWatch Database
Insights 確認

ここを忘れずに!

インスタンス最適化

1. インスタンス
ダウンサイジング

2. インスタンスタイプ
変更

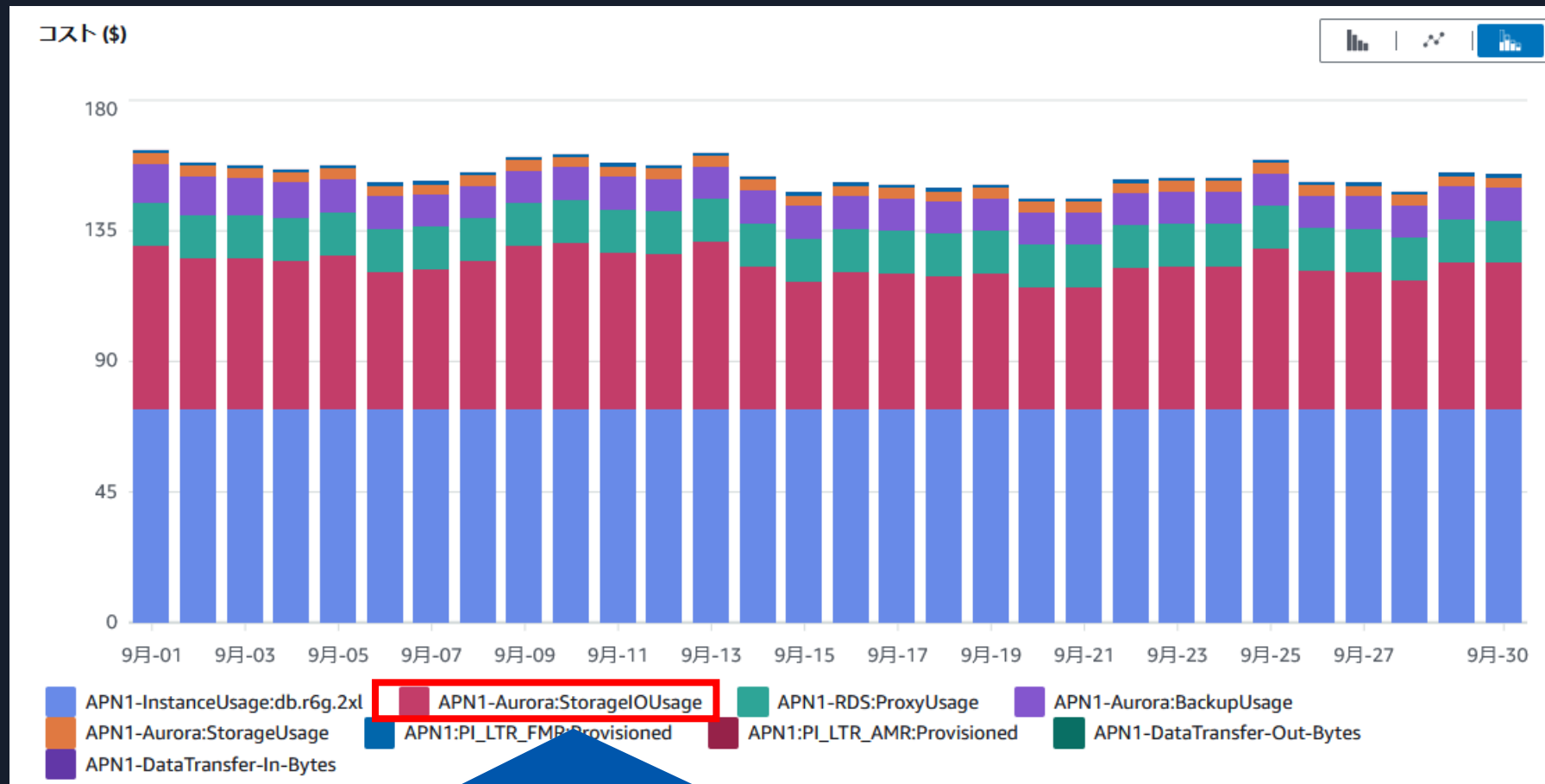
RI 購入／クラスタ停止

3. Reserved Instance
購入

4. 利用しない
クラスタ停止

Amazon Aurora I/O 最適化

I/O 最適化インスタンス検討の契機



日付範囲
2025-09-01 — 2025-09-30

粒度
日別

▼ グループ化の条件
ディメンション クリア
使用タイプ

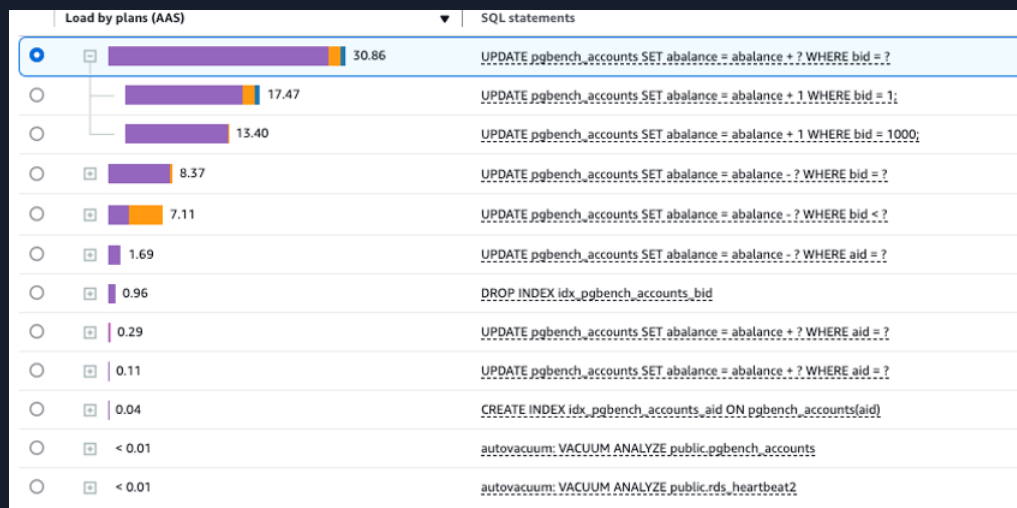
▼ フィルター 情報
適用フィルター (2) [すべてをクリア](#)
サービス クリア
サービスを含む (1)
Relational Database Service (RDS)

Aurora コストの I/O 利用料の比率が高い場合、I/O 最適化インスタンスを検討する

精益求精 (せいえききゅうせい)

すでに優れているものでも、さらなる改善を追求するという考え方

I/O を減らす事が出来ないか、確認しましょう



I/O 最適化インスタンスの概要

クラスターストレージ設定 情報

アプリケーションの料金予測可能性と料金パフォーマンスのニーズに最適な Aurora DB クラスターのストレージ設定を選択します。

設定オプション

データベースインスタンス、ストレージ、I/O の料金は、設定によって異なります。詳細はこちら [🔗](#)

☒ Aurora I/O 最適化

- すべてのアプリケーションの予測可能な料金設定。I/O の多い (I/O コスト > データベース全体のコストの 25%) アプリケーションの料金パフォーマンスが向上。
- 読み取り/書き込み I/O 操作には追加料金はかかりません。DB インスタンスとストレージの料金には I/O 使用量が含まれています。

☐ Aurora スタンダード

- I/O 使用量が中程度 (I/O コスト < 25% of total database costs).) の多くのアプリケーション向けの費用対効果の高い料金設定
- リクエストごとの支払い I/O 料金が適用されます。DB インスタンスとストレージの料金には I/O 使用量は含まれていません。

- お客様はコンピューティングとストレージのみで、**読み取り/書き込み IO にコストがかからない**
- Aurora の多様なワークロードにおいて**コストの予測が容易に**
- Aurora クラスターは **30 日に 1 回、ストレージオプション (スタンダード → I/O 最適化) を変更**する事が可能

Aurora I/O 最適化 のコスト

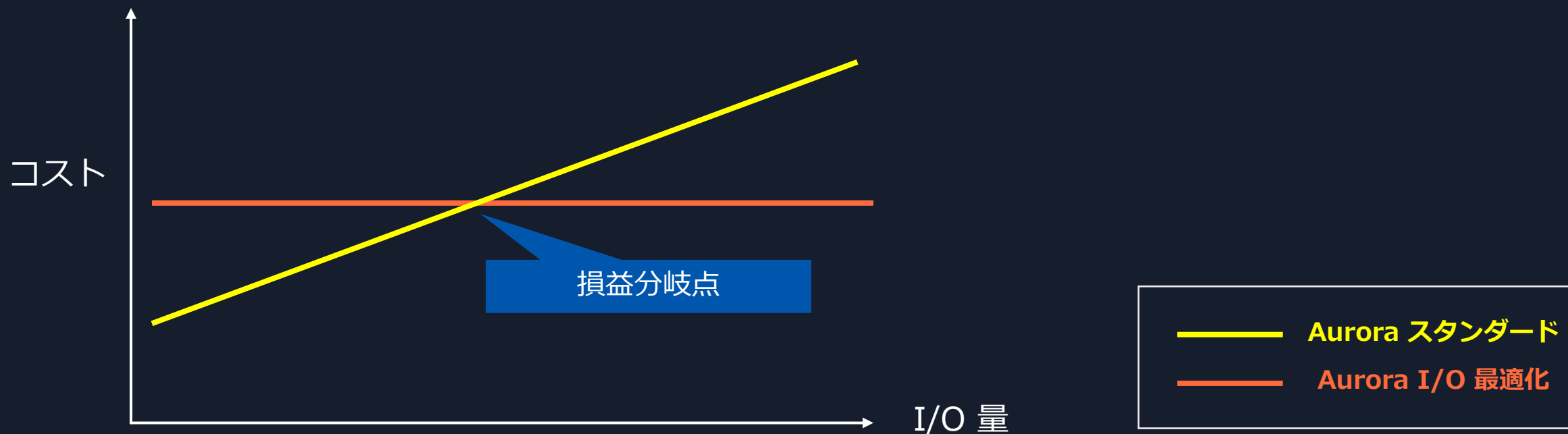
東京リージョンにおける I/O 最適化とスタンダードのコスト比較 (2025 年 10 月現在)

ストレージ設定	コストカテゴリ	費用
Aurora スタンダード	インスタンス	db.r5.xlarge : \$ 0.70/時間
	ストレージ	\$ 0.12/毎月の GBあたり
	I/O	\$ 0.24/100万リクエスト
Aurora I/O 最適化	インスタンス	db.r5.xlarge : \$ 0.91/時間 Aurora スタンダードの +30%
	ストレージ	\$ 0.27/毎月の GBあたり Aurora スタンダードの +125%
	I/O	利用料に含まれる

Aurora I/O 最適化 の損益分岐点

I/O コストが Aurora データベースの総コストの **25%** を超えている場合がざっくりとした損益分岐点

コストと I/O 量の関係図 (イメージ)



Amazon Aurora Serverless V2



Aurora Serverless v2 の特徴

インスタンスの設定

以下の DB インスタンスの設定オプションは、上記で選択したエンジンでサポートされているものに制限されています。

DB インスタンスクラス | [情報](#)

▼ フィルターの非表示

☐ 以前の世代のクラスを含める

☒ Serverless v2

☐ メモリ最適化クラス (r クラスを含む)

☐ パースト可能クラス (t クラスを含む)

☐ 最適化された読み取りクラス - 新規

Serverless v2

最も要求の厳しいワークロードでも瞬時にスケーリング。

容量の範囲 | [情報](#)

データベース容量は Aurora 容量ユニット (ACU) で測定されます。1 ACU は 2 GiB のメモリと、対応するコンピューティングとネットワーキングを提供します。

最小キャパシティ (ACU)

0

(0 GiB)

0 ~ 256 (0.5 の増分)

最大キャパシティ (ACU)

64

(128 GiB)

1 ~ 256 (0.5 の増分)

非アクティブ後に一時停止 | [情報](#)

最小キャパシティ設定が 0 ACU のクラスターは、非アクティブの間は一時停止できます。DB インスタンスがアイドル状態になれる時間 (一時停止するまでの時間) を指定します。詳細については、[Aurora Serverless DB インスタンスを一時停止](#) を参照してください

00:05:00

5 分から 24 時間までの時間を入力してください

- インスタンスのリソース容量を **ACU (Aurora Capacity Unit)** で管理
- 各 ACU は**約 2GiB(ギビバイト) のメモリと対応する CPU**、ネットワークが組み合わされた容量となる
- 最小 ACU と 最大 ACU を指定し、その範囲で**自動的にスケールアップ／スケールダウン**を実施する
- ACU は **0 ~ 256** の範囲で指定可能
- 使用頻度の低いアプリケーション、開発・テスト用のデータベース、可変・予測不可能なワークロードに最適
- コストの正確な予測が難しい点は注意

Aurora Serverless V2 のインスタンス設定

Aurora Serverless v2 のコスト試算

ピーク時のみスケールするようなワークロードに対して、Serverless V2 を利用することでコスト最適化が見込める

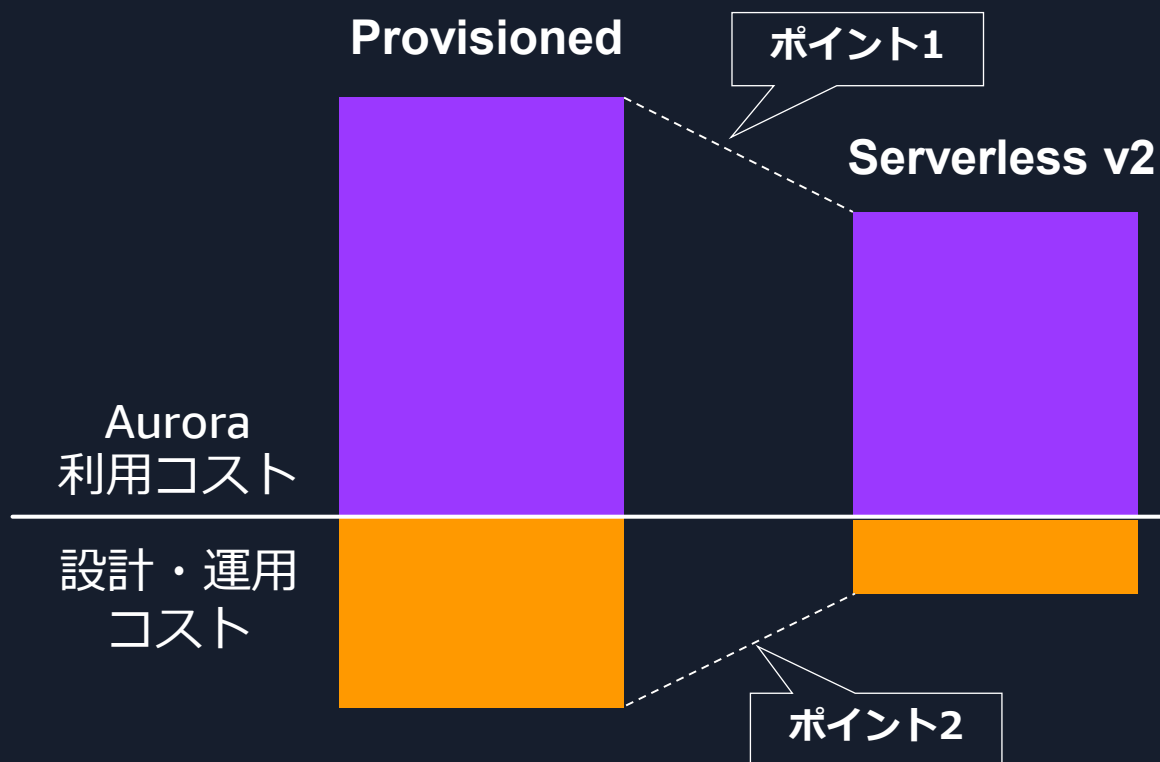
種類		スペック	インスタンスタイプ 別料金 (*2)	料金 (30日換算)
Provisioned	Amazon Aurora	4vCPU/32GiB memory	db.r7i.xlarge \$ 0.70/時間	\$ 504.00
Serverless	Aurora Serverless V2	2 ACU (4GB) ~ 16 ACU (32GiB) *1	下記ピーク時は 16 ACU (通常時は 2 ACU) 毎日 11:00~13:00 (2h) \$ 0.15/ACU 時間	\$ 342.00
		0 ACU ~ 16 ACU (32GiB)	下記ピーク時は 16 ACU (通常時は 0 ACU) 毎日 11:00~13:00 (2h) \$ 0.15/ACU 時間	\$ 144.00

*1 - Performance Insights 利用時における推奨の ACU 最小容量は 2 ACU

*2 - 2025 年現在の Aurora Standard 東京リージョンの価格

<https://aws.amazon.com/jp/rds/aurora/pricing/>

Aurora Serverless v2 導入を検討するポイント



ポイント1

- ワークロードによっては Provisioned の方がコストメリットがある場合もある
- ワークロードの特性を見極めた上で、Serverless v2 利用によるコストメリットの有無を確認**する
- 机上でのワークロード特性の見極めが困難な場合は、PoC を実施した確認が有効

ポイント2

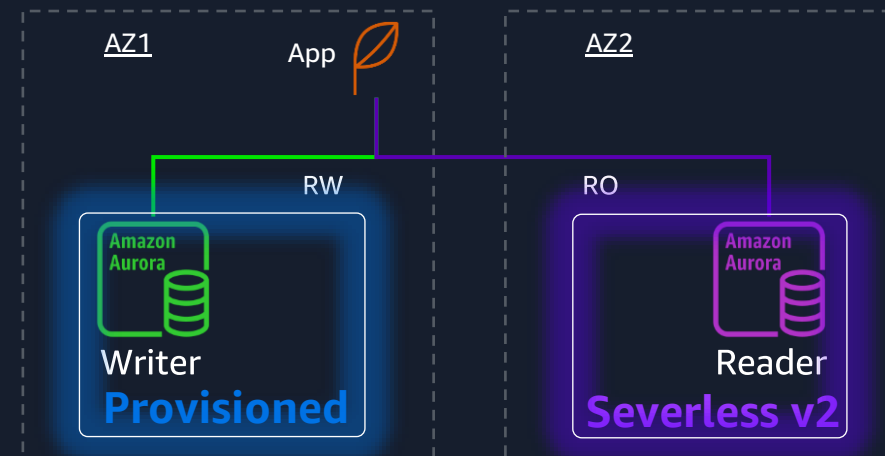
- Serverless v2 により、**データベースインスタンスのキャパシティ管理コストを抑制**することができるため、どの程度の設計・運用コストが削減できるか？という観点で検討する
- 設計・運用コストだけでなく、人的リソースにおける負荷軽減という観点でも検討する価値あり

* 上記図は実際のコストの比率を図示したものではありません。イメージとして捉えてください。

Aurora Serverless v2 の混在環境での利用

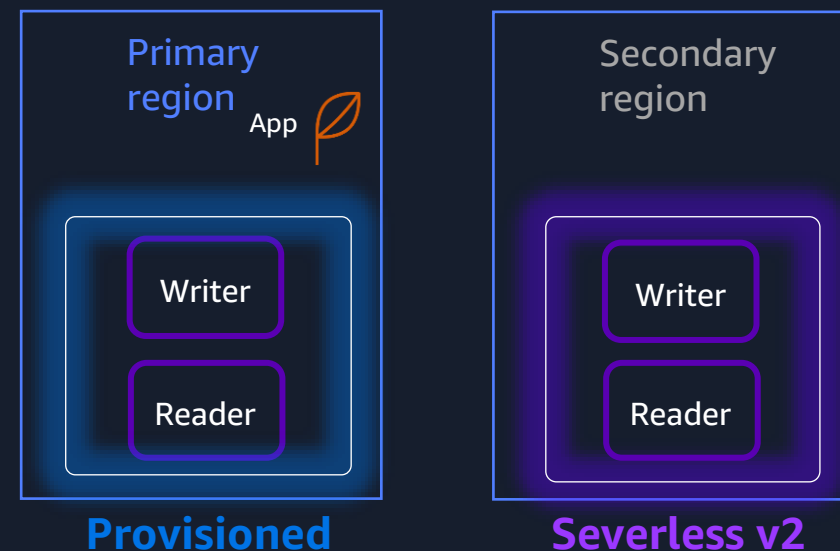
リーダーインスタンスを Serverless v2 に

- Writer のキャパシティは Provisioned で運用
- Reader のキャパシティは Serverless V2 で運用し、**Reader がアイドル時は最小キャパシティのみの支払い** ※に留める (※ priority-tier : 2 ~ 15 の場合)



セカンダリリージョンを Serverless v2 に

- Aurora Global Database のセカンダリリージョンを Aurora Serverless V2 で構成
- **セカンダリリージョンがアイドル時は最小キャパシティ分のみの支払い**に留める
- 従来のヘッドレス構成より、**オペレーションが自動化され、サービス再開が高速化**出来る



まとめ

- Amazon Aurora のコスト構造
→ **Cost Explorer** でどこにコストがかかっているか把握しましょう
- Amazon Aurora インスタンス最適化
→ **ワークロードにあったインスタンスタイプ**を選びましょう
- Amazon Aurora I/O 最適化
→ **I/O 費用が 25% を超えるような場合、最大で 40% のコスト削減に**
- Amazon Aurora Serverless V2
→ **予測不可能なワークロードに最適、混在環境にも利用可能**

Thank you!