JAPAN | 2024

aws SUMMIT

AWS-38

NoSQL コスト削減大全

Mastering Cost Optimization for NoSQL

堤 勇人(Tsutsumi, Hayato)
アマゾン ウェブ サービス ジャパン 合同会社
Senior In-memory Specialist Solutions Architect

自己紹介

堤 勇人 (Tsutsumi, Hayato)

Sr. In-memory Specialist Solutions Architect

前職はパッケージシステム開発者

趣味：スキー、街歩き、ゲーム

好きな AWS のサービス：Amazon ElastiCache, Amazon MemoryDB for Redis, Amazon DynamoDB

Agenda

- 何故 NoSQL のコスト削減は有効なのか
- NoSQL コスト削減ベストプラクティス
- 持続的なコスト削減に向けて
- まとめ

何故NoSQLのコスト削減は有効なのか

コストとは？

インフラコスト

ライセンスコスト

オペレーションコスト

開発コスト

ビジネスコスト

データベースのコスト削減は難しい？

- コストが大きい部分の一つ
- 高い可用性が求められる
- 専門性が高い

↓

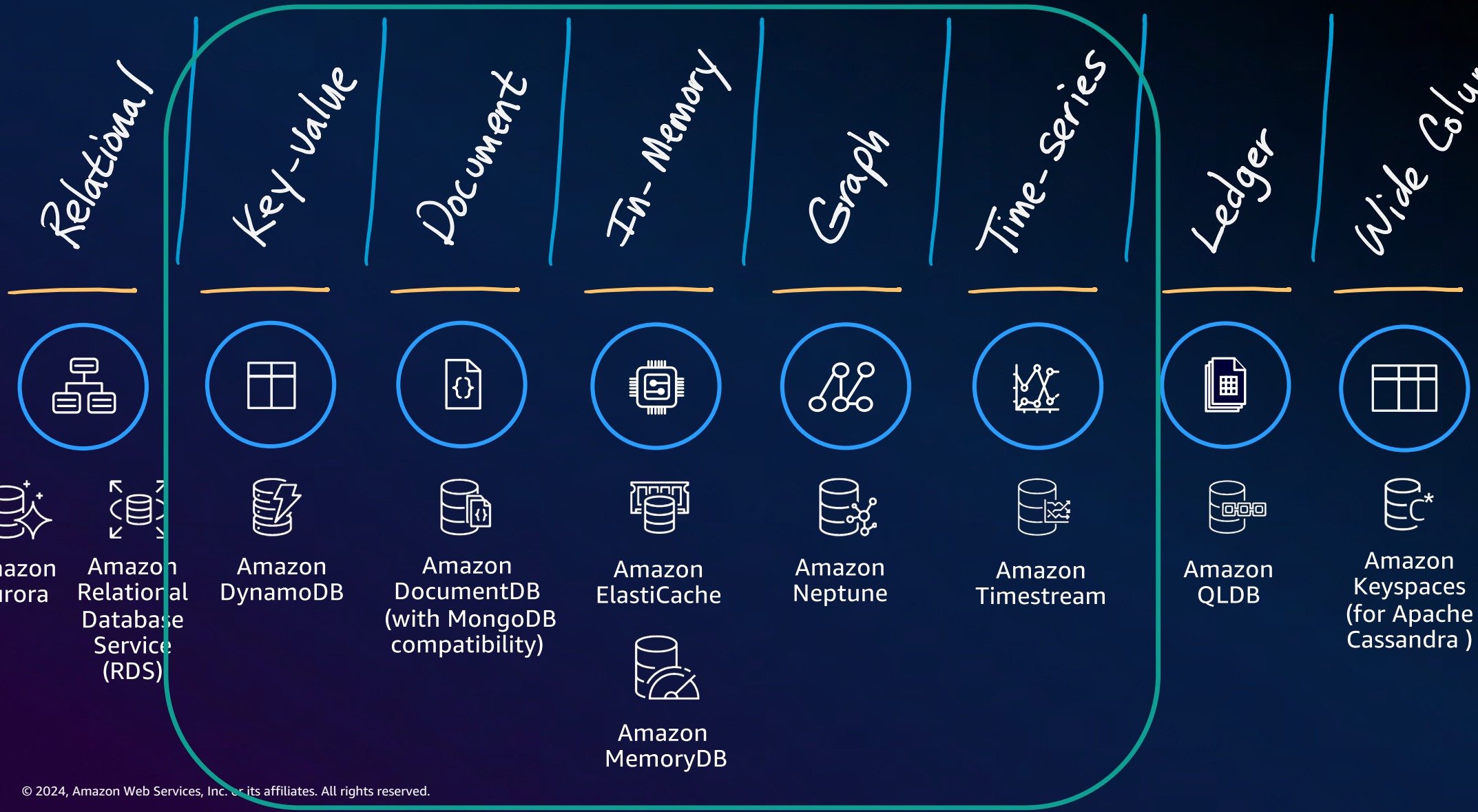
”聖域”とされてしまうことも

NoSQL のコスト削減

- NoSQL のコスト削減は比較的容易
- AWS のデータベースサービス
  = Purpose-built databases
- 機能が目的別に特化しているため、
  ベストプラクティスが作りやすい
- 逆に目的に沿わない使い方はアンチパターンとなる

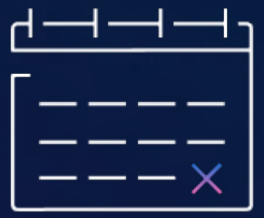
NoSQL コスト削減 ベストプラクティス

Purpose-built databases 今回お話しする範囲



NoSQLにおけるコスト削減ポイント

サイジング



スケジューリング

価格体系

オプション設定

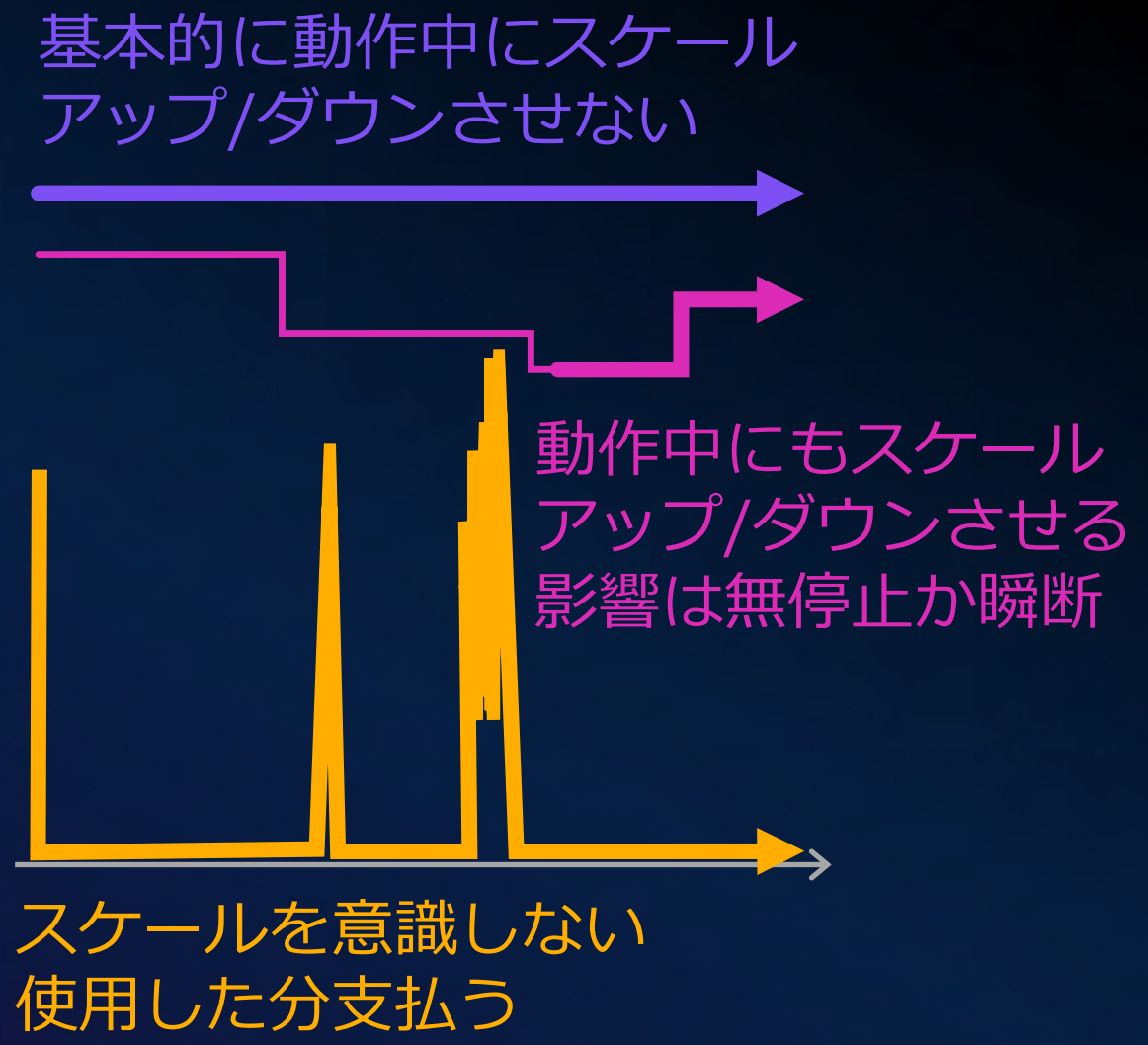
データモデリング

コンピューティングリソースの価格体系

名称	プロビジョンド インスタンス	プロビジョンド キャパシティ	オンデマンド キャパシティ
タイプ	インスタンス	サーバーレス	サーバーレス
動的なスケール操作	スケールイン・アウト	キャパシティ変更	キャパシティ変更
支払い	割り当てた分	割り当てた分	実際に利用した分
適するワークロード	定常的なワークロード	ある程度予測可能で変化が滑らかなワークロード	予測不可能、もしくは変化が急激なワークロード

コスト削減ポイント - 価格体系詳細

- コンピューティングリソース
  - プロビジョンドインスタンス
  - プロビジョンドキャパシティ
  - オンデマンドキャパシティ
  - リザーブドは適用できるか
- 適切なオプションを選ぶ
  - I/O, メモリ, ストレージ, バックアップ
- 価格設定の最低値も見る



Amazon DynamoDB

- 一桁ミリ秒の応答速度
- 大規模なスケーラビリティ
- サーバーレス

大規模なパフォーマンス

安全性と耐障害性

サーバーレス

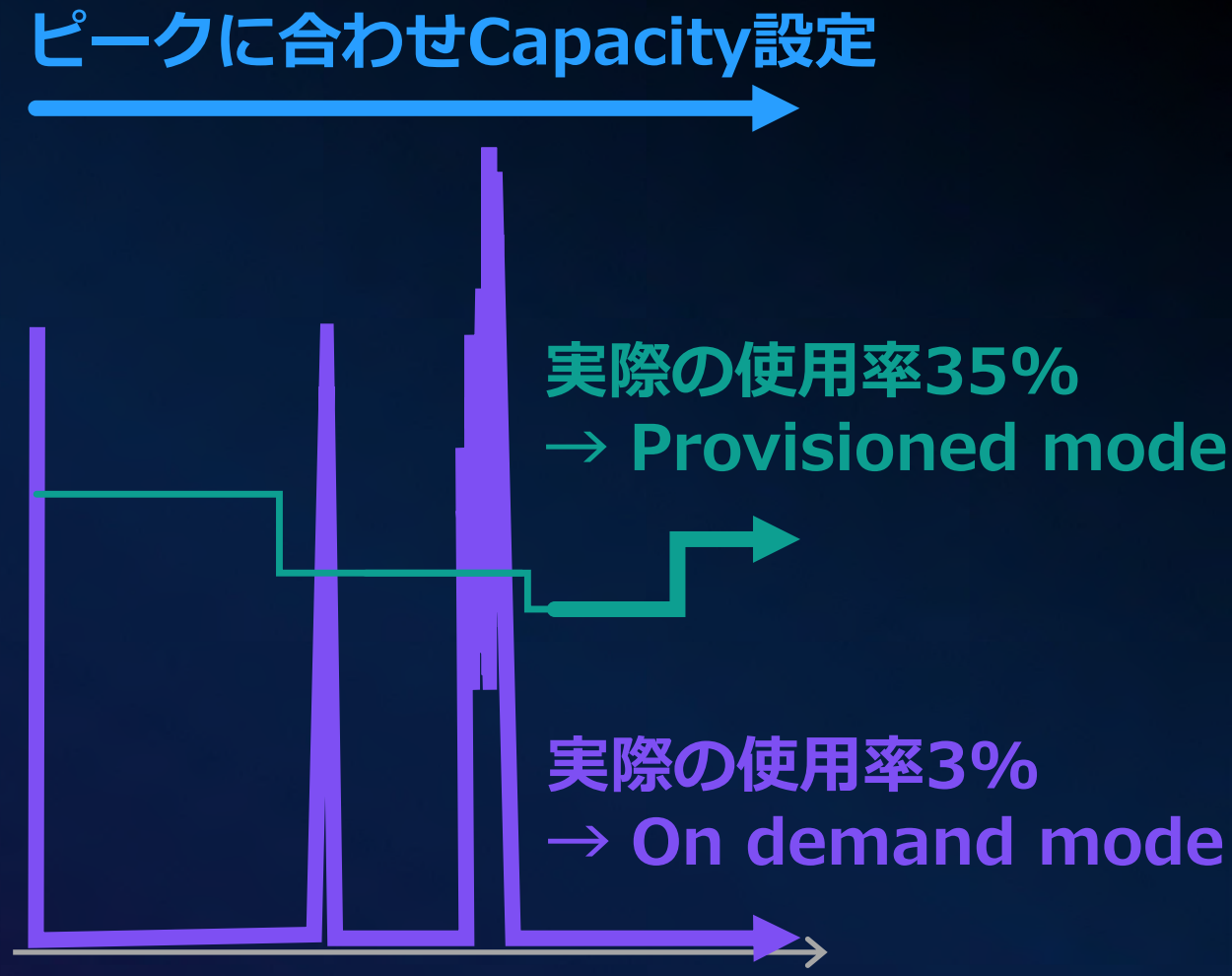
他の AWS サービスとの統合

Amazon DynamoDB

- 価格体系
  - プロビジョンドキャパシティ = Provisioned mode
  - オンデマンドキャパシティ = On-demand mode
- 読み込みは4KB毎、書き込みは1KB毎
- 結果整合性のある読み込みは消費半分
- 読み込みは書き込みの5倍安い

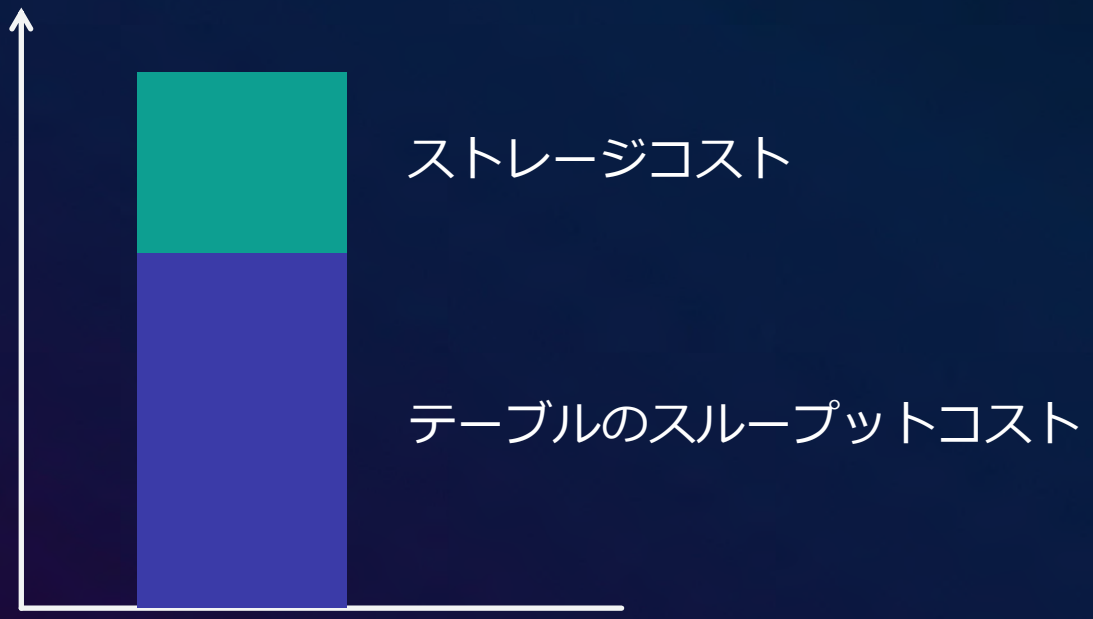
Amazon DynamoDB

- Provisioned mode は On-demand mode の最大7倍安い
  - オートスケーリングで使用率が 14.4%であることと同値
- さらに Provisioned mode はリザーブドキャパシティが使える
  - 最大77%減
- 一方、オートスケーリングの設定や ピーク時の処理等は必要
  - オペレーションコストに繋がる



Amazon DynamoDB

- ストレージ書き込みが多い場合は Standard-IA クラスを使う
  - DynamoDB 標準テーブルクラスを使用時、テーブルのスループット コストの 50% をストレージコストが超える場合、メリットがある



Amazon DocumentDB

- 非構造型データ (JSON) の格納
- 複雑なクエリ、アドホックなクエリが使える

{ }

flexible JSON format

Modeling in JSON

API

common API output

高速でスケーラブル

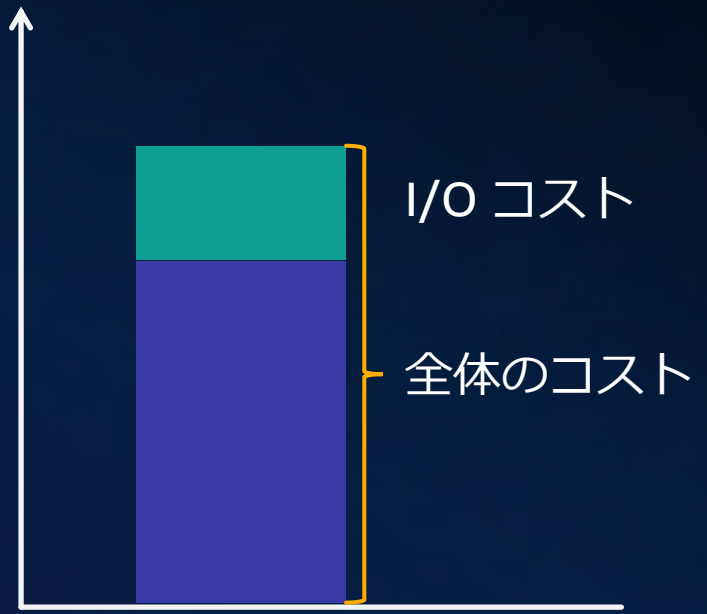
フルマネージド

エンタープライズ準拠の機能

MongoDB 互換

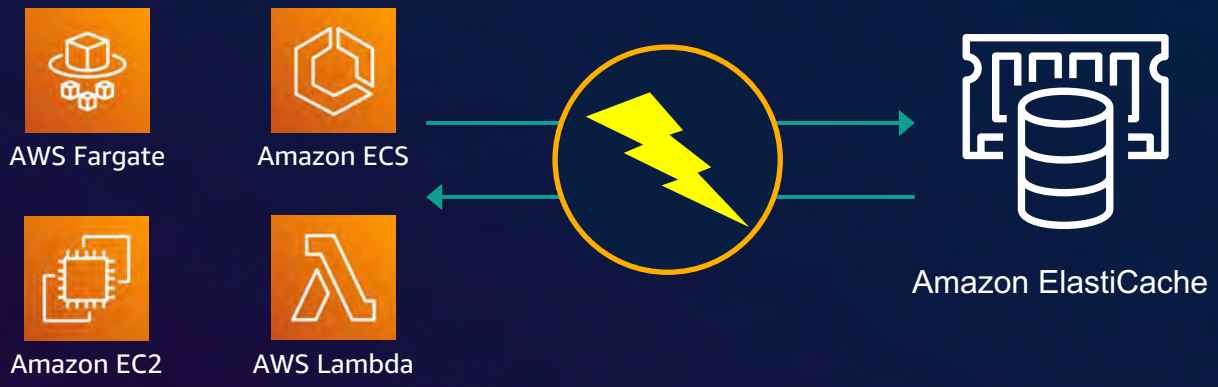
Amazon DocumentDB

- 価格体系
  - プロビジョンドインスタンス
  - プロビジョンドキャパシティ = ElasticCluster
- スケジューリング
  - クラスターを一時的に停止する
  - 最大7日間停止できる
- I/O 最適化ストレージオプション
  - I/O コストがデータベースの総コストの 25% を超える
- TTL ストリームはI/Oを消費する
  - ローリングコレクションでコレクションごと消す



Amazon ElastiCache

- マイクロ秒レベルの応答速度
- Redis & Memcached 互換
- サーバーレス対応



Redis &
Memcached 互換

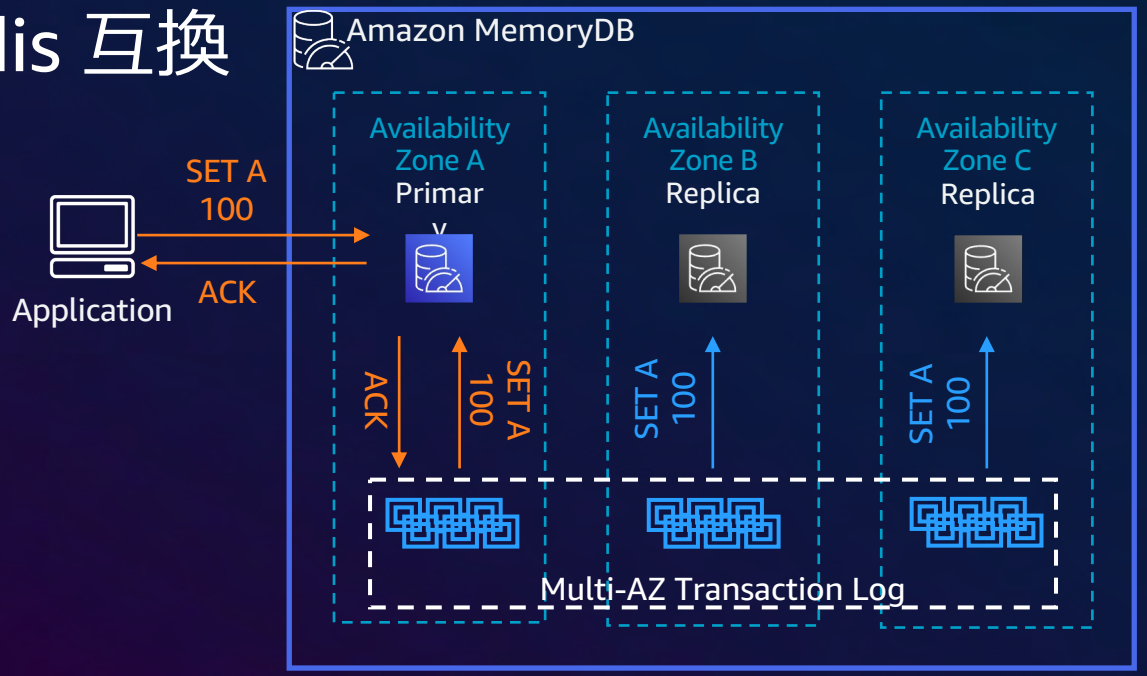
超高速なパフォーマンス

セキュア

フルマネージド & サーバーレス

Amazon MemoryDB

- インメモリの応答速度
- プライマリデータベースとしての
  マルチ AZ 耐久性
- Redis 互換



超高速
パフォーマンス

フルマネージド

Redis 互換

セキュリティと統制

スケーラビリティ

耐久性と可用性

Amazon ElastiCache / Amazon MemoryDB

- 価格体系
  - プロビジョンドインスタンス
  - オンデマンドキャパシティ = ElastiCache Serverless
- Reserved Nodes は最大55%削減
- バージョンアップ
- データ階層化インスタンスタイプ
  - データが非常に多く、全体の20%以下に頻繁にアクセスされるユースケース
  - マッチするユースケースでは60%以上のコスト削減

ElastiCache performance improvement history

2019/03

Redis 5.0.3
Enhanced I/O Processing
(up to 83% improved throughput
and 47% latency)

2022/08

Graviton2 R6g/M6g
(up to 45% improved
performance)

2023/01

Redis 7.0.5
Enhanced I/O multiplexing
(up to 72% improved throughput
and 71% P99 latency)

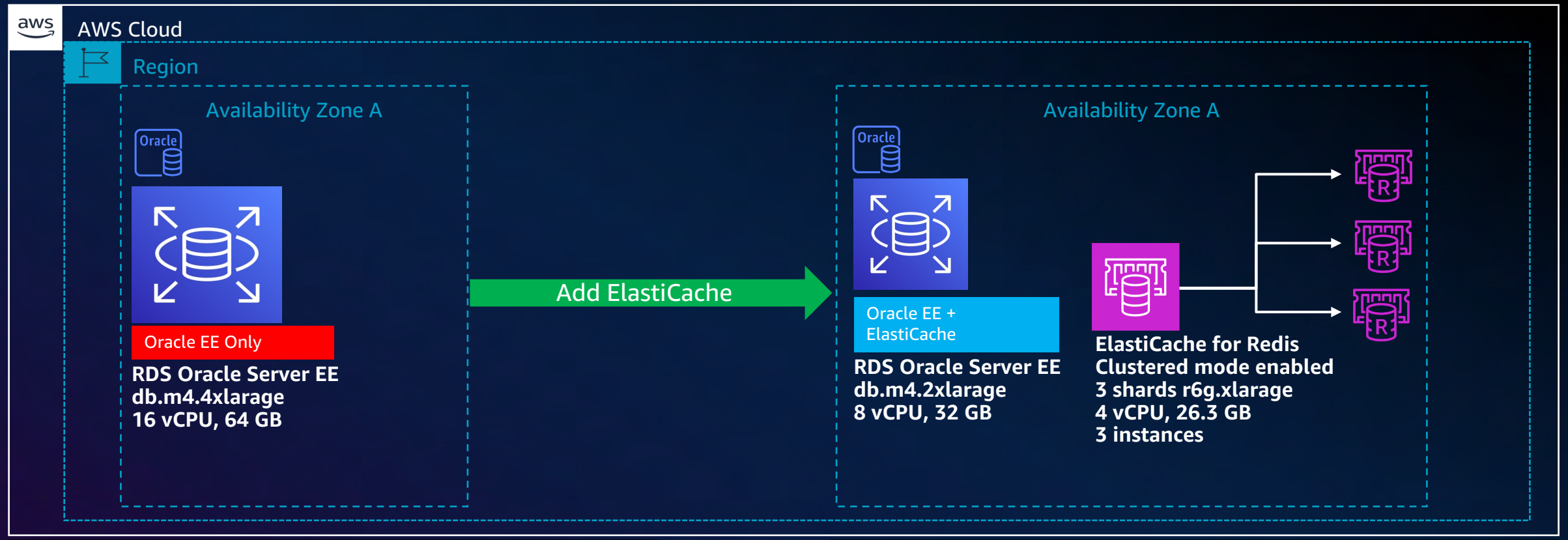
2023/08

Graviton3 R7g/M7g
(up to 28% improved throughput
and 21% P99 latency)

2023/11

Redis 7.1
Enhanced I/O
presentation layer offloading
(up to 100% more improved throughput
and 50% lower P99 latency)

他のサービスのコスト削減 - キャッシュレイヤ

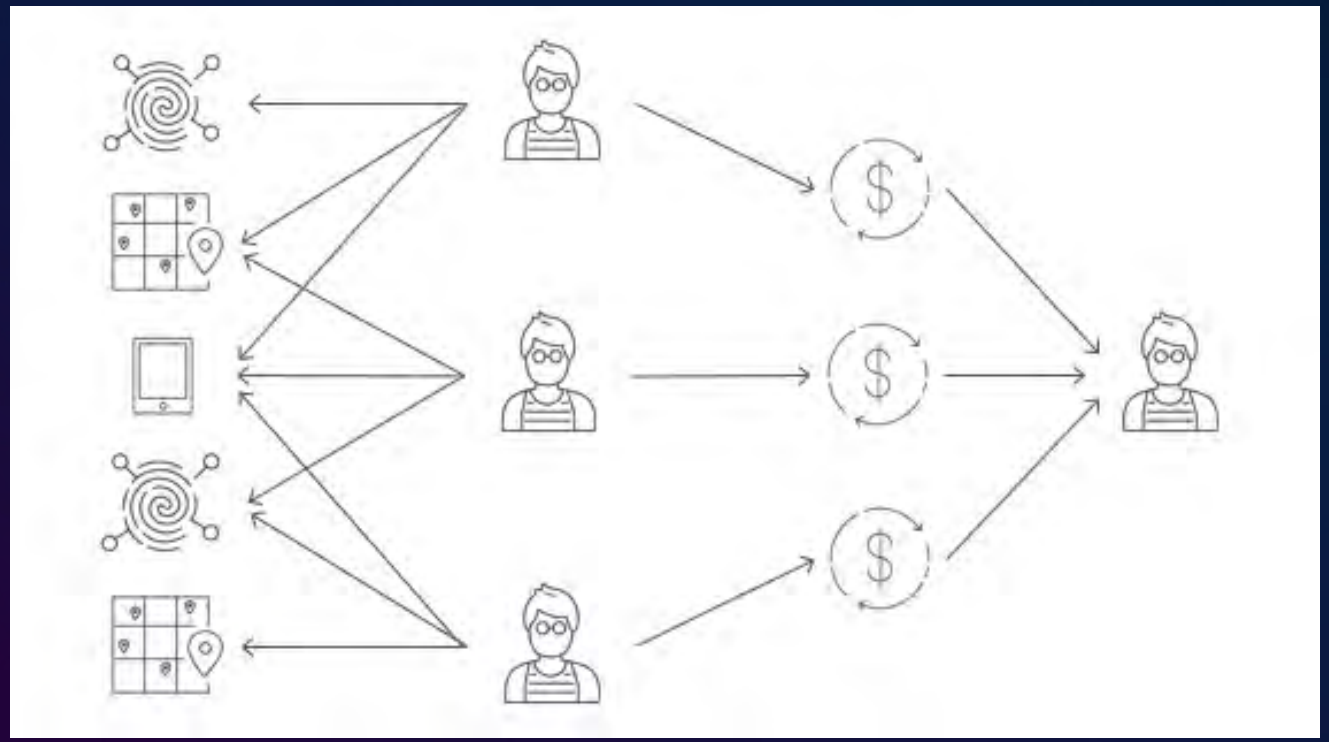


Original Configuration	vCores	RAM	Annual Cost
RDS Oracle EE (m4.4xlarge)	16	64	$57,917
Total	**16**	**64**	**$57,917**

New Configuration	vCores	RAM	Annual Cost
RDS Oracle Server (m4.2xlarge)	8	32	$29,096
ElastiCache (r6g.xlarge)x3	12	78	$10,801
Total	20	110	**$39,896**
Savings			**~31%**

Amazon Neptune Database / Analytics

- グラフ構造に対して効率的にクエリ
- OLTP だけでなく OLAP, 機械学習にも対応



スケーラブル

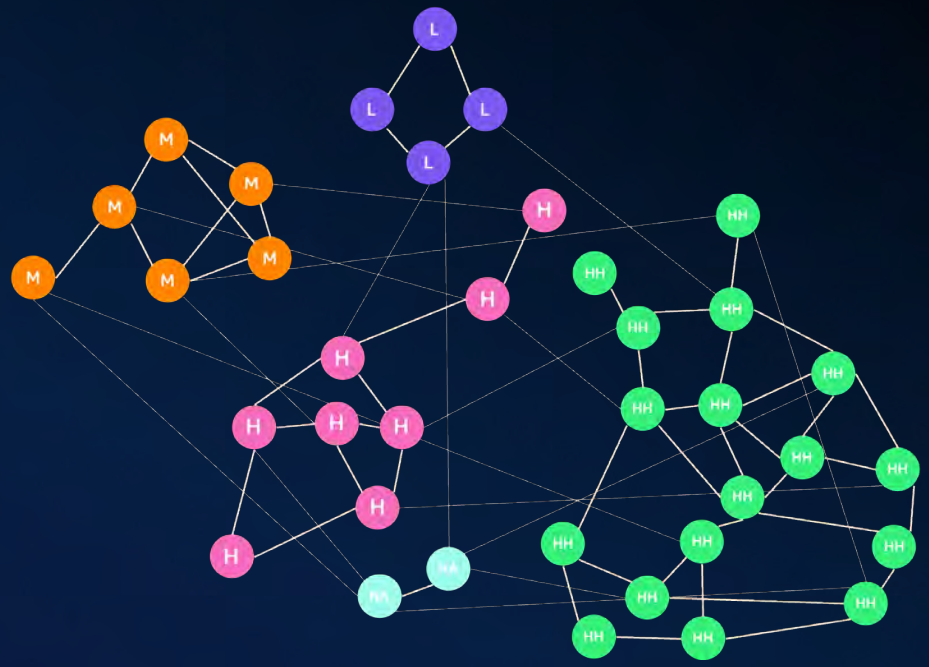
セキュア・信頼性

サーバーレス

インテグレーション

Amazon Neptune Database / Analytics

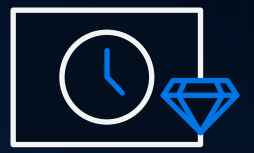
- 価格体系
  - プロビジョンドインスタンス
  - プロビジョンドキャパシティ
- スケジューリング
- バージョンアップ
- クエリチューニング
- Spark GraphX からの移行



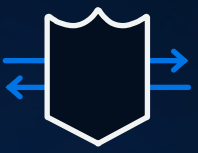
Amazon Timestream

- 時系列データの扱いに特化
- LiveAnalytics に加え、Managedサービスとして InfluxDB 版をリリース

パフォーマンスと
スケーラビリティ



可用性と耐久性



セキュリティ

SQLで操作可能な
時系列関数

Amazon Timestream

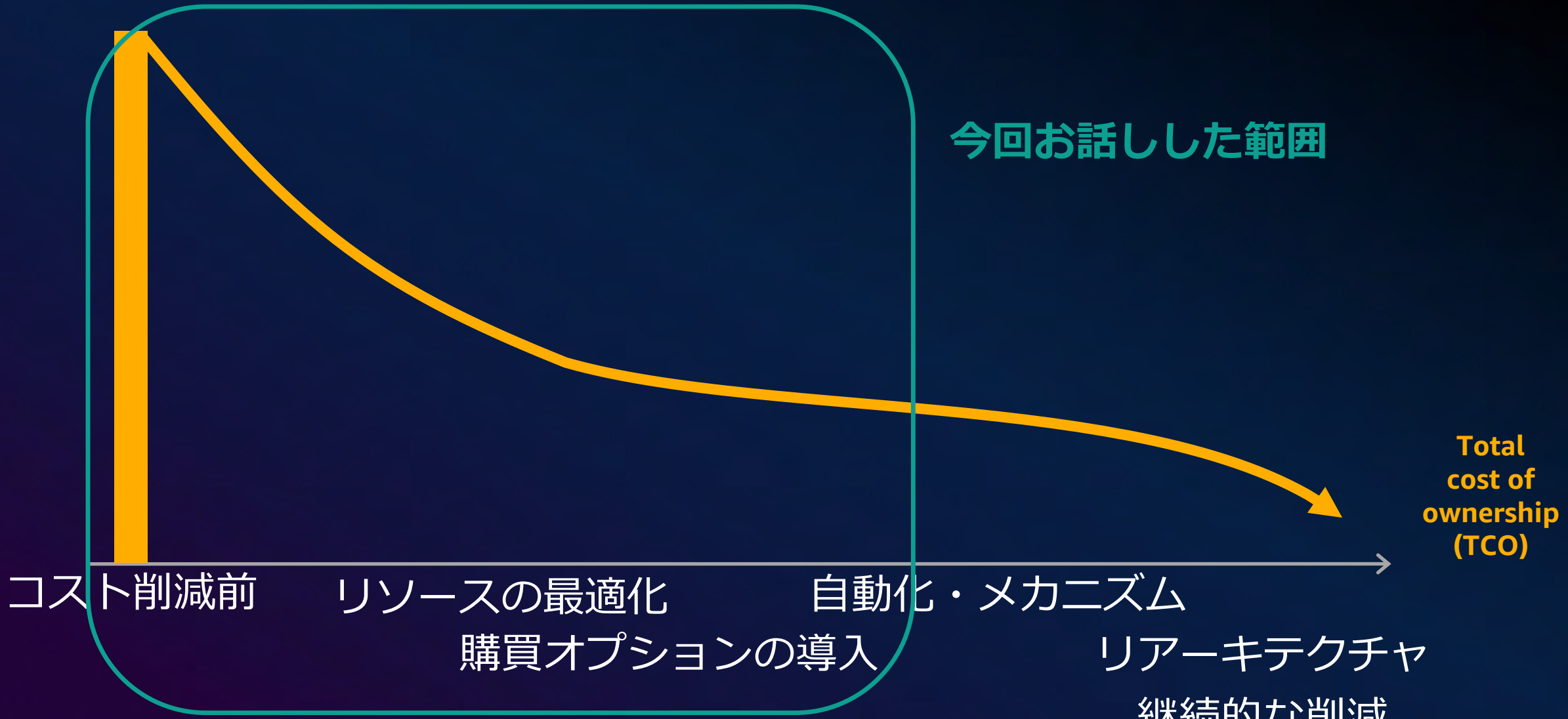
- 課金体系
  - プロビジョンドインスタンス = InfluxDB
  - オンデマンドキャパシティ = LiveAnalytics
- コスト削減アプローチ
  - 必要なデータのみを Bulk で書き込む
  - メモリ層の保存期間を適切に設定
  - クエリチューニング
- 適切なエンジン選択

Amazon Timestream
for InfluxDB

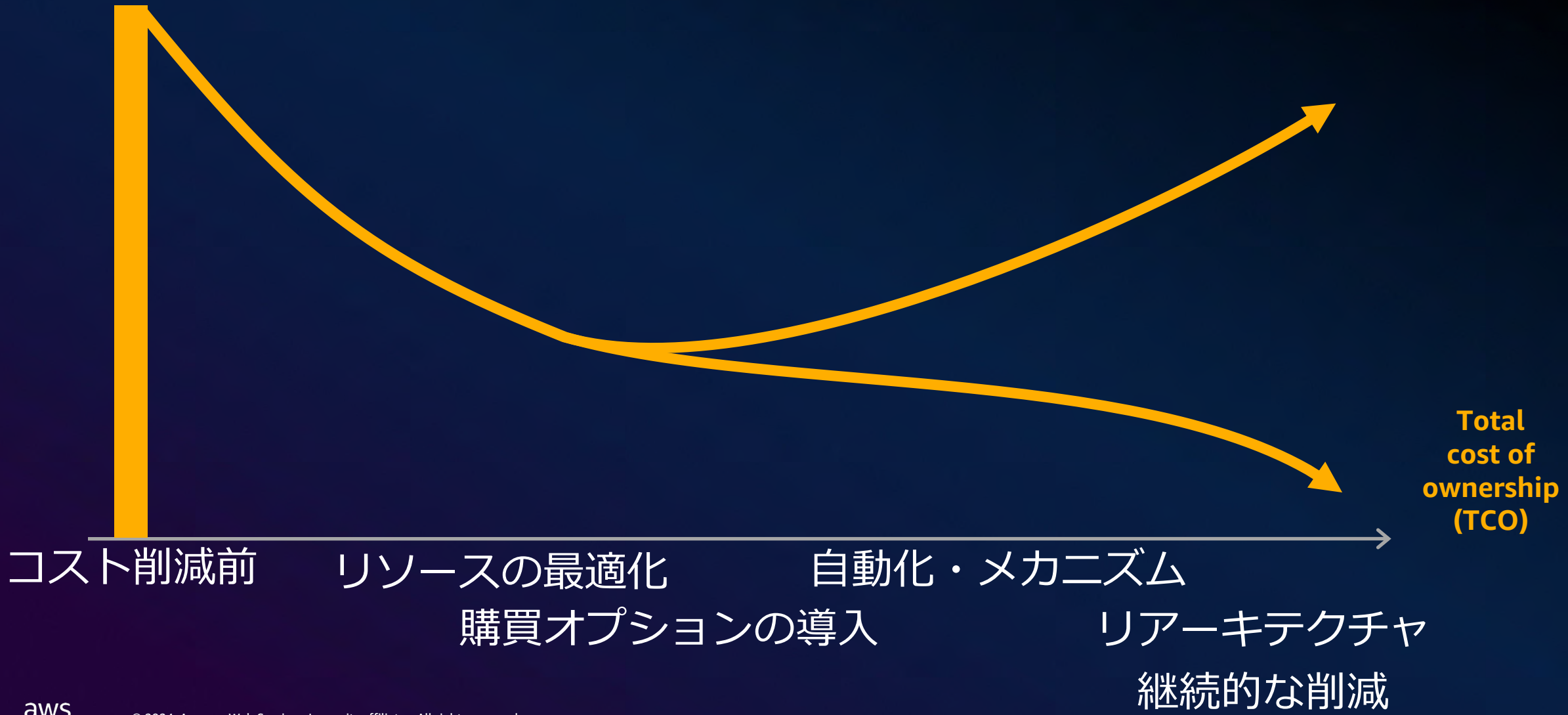
Amazon Timestream
for LiveAnalytics

持続的なコスト削減に向けて

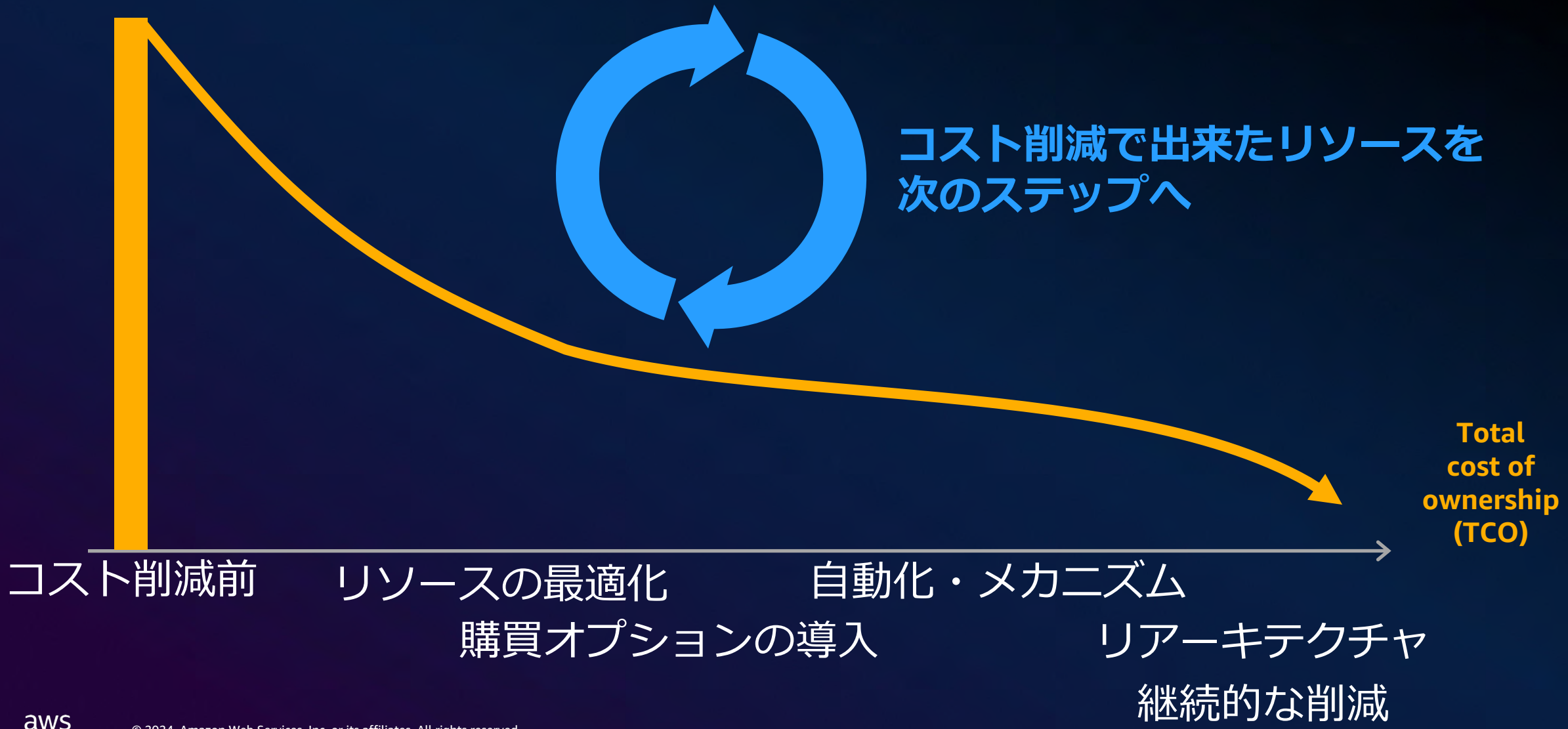
コスト削減のステップ



クイックウィンで満足してはいけない

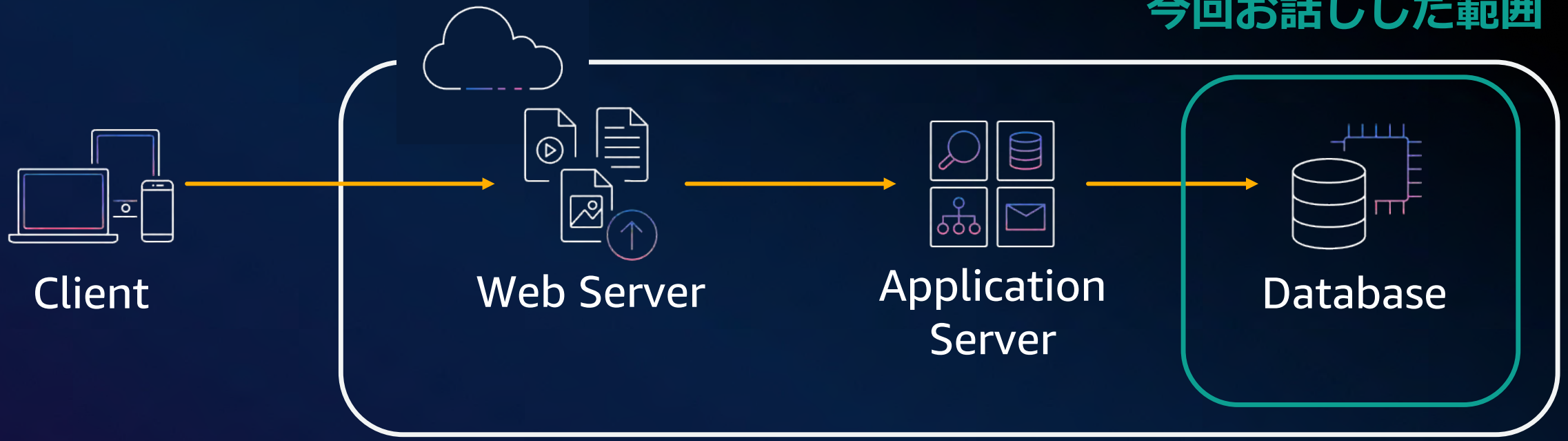


コスト削減のステップ



組織的なコスト最適化

今回お話しした範囲



Organization

関連機能・情報

- AWS Cost Explorer コスト最適化ハブ
- AWS Compute Optimizer
- Trusted Adviser
- AWS Well-Architected Framework
  - Database Lens

Link:
https://us-east-1.console.aws.amazon.com/costmanagement/home?region=ap-northeast-1#/cost-optimization-hub
https://us-east-1.console.aws.amazon.com/compute-optimizer/home?region=ap-northeast-1#/dashboard
https://us-east-1.console.aws.amazon.com/trustedadvisor/home?region=ap-northeast-1#/category/cost-optimizing
https://aws.amazon.com/jp/architecture/well-architected/

まとめ

- NoSQL (Purpose built database) は比較的コスト削減しやすい
  - データベースの設計思想通りに使うことがベストプラクティス
- データベースレイヤーであってもリファクタリングの前にできることがある
- 見るべきポイントはサイジング、スケジューリング、価格体系
- クイックウィンに満足せず、コスト削減を継続する組織作りが必要

Thank you!