第31回 ちょっぴり DD

Amazon Aurora Serverless v2 移行への道

2023.06.29

自己紹介

鈴木康寛

クリエイティブサーベイ株式会社

CTO

2013年にSansan入社。
webアプリケーションエンジニアとして、Eightの開発に従事。サーバーレスアーキテクチャを用いた、リコメンデーションサービスに関して、2017 AWS Dev Dayにて登壇。
その後、エンジニアリングマネージャーの経験を経て、昨年6月より現職。

CREATIVE SURVEY

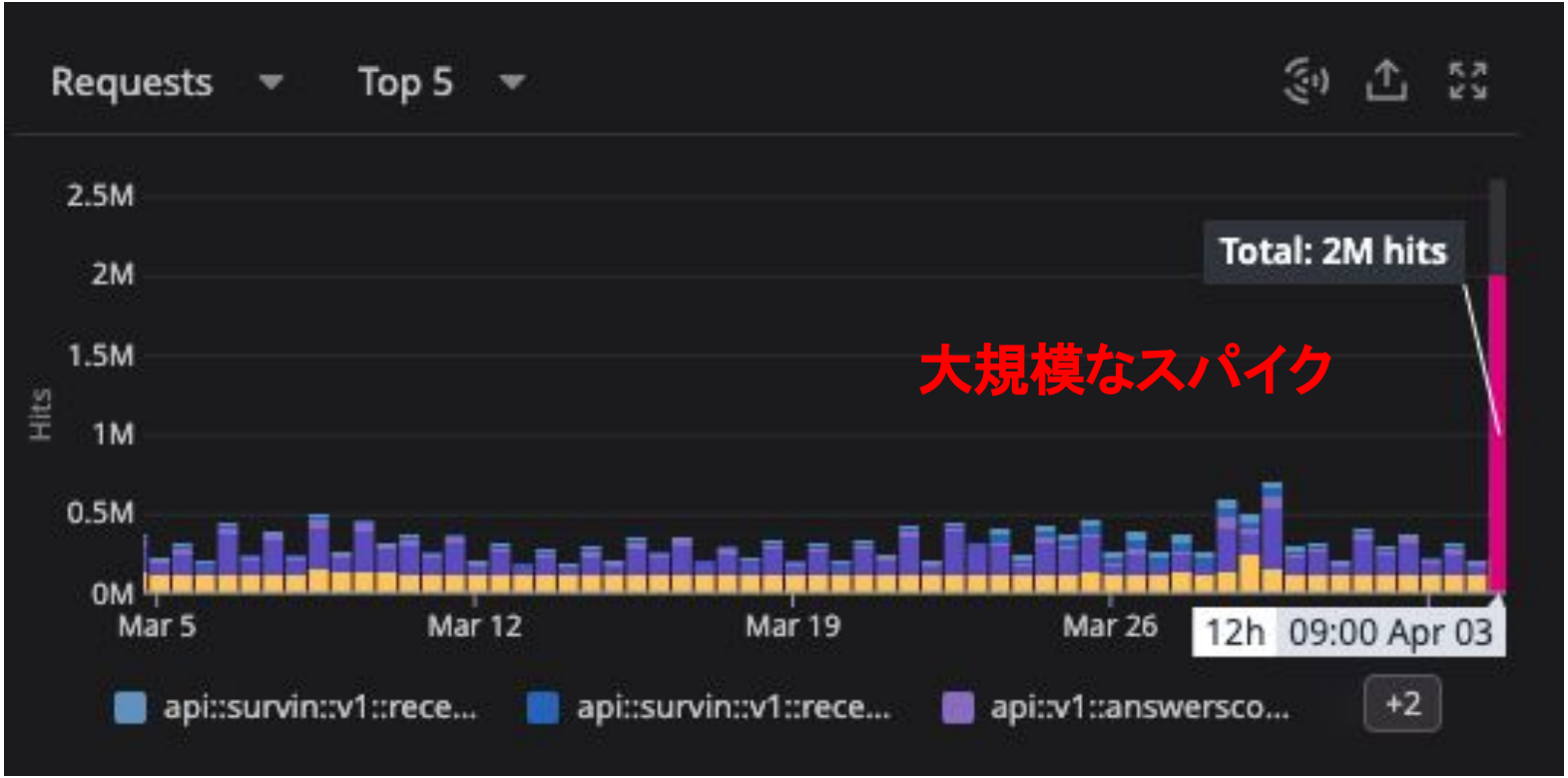
今年よりSansanグループに参画

sansan × CREATIVE SURVEY

ミッション

顧客の声を機会に変える

アンケートサービスの特性



スケーラビリティとの戦い

2022/6: CTO就任

2022/6〜: 諸々の技術的負債への対応

- Ruby/Rails/MySQLバージョンアップ
- 開発環境のコンテナ化
- CI / CDパイプライン整備によるデプロイの自動化

2022/12: EC2ベースからECS on Fargateベースになり、オートスケーリングが可能に

データベースの変遷

2022/6～: MySQLバージョンアップ

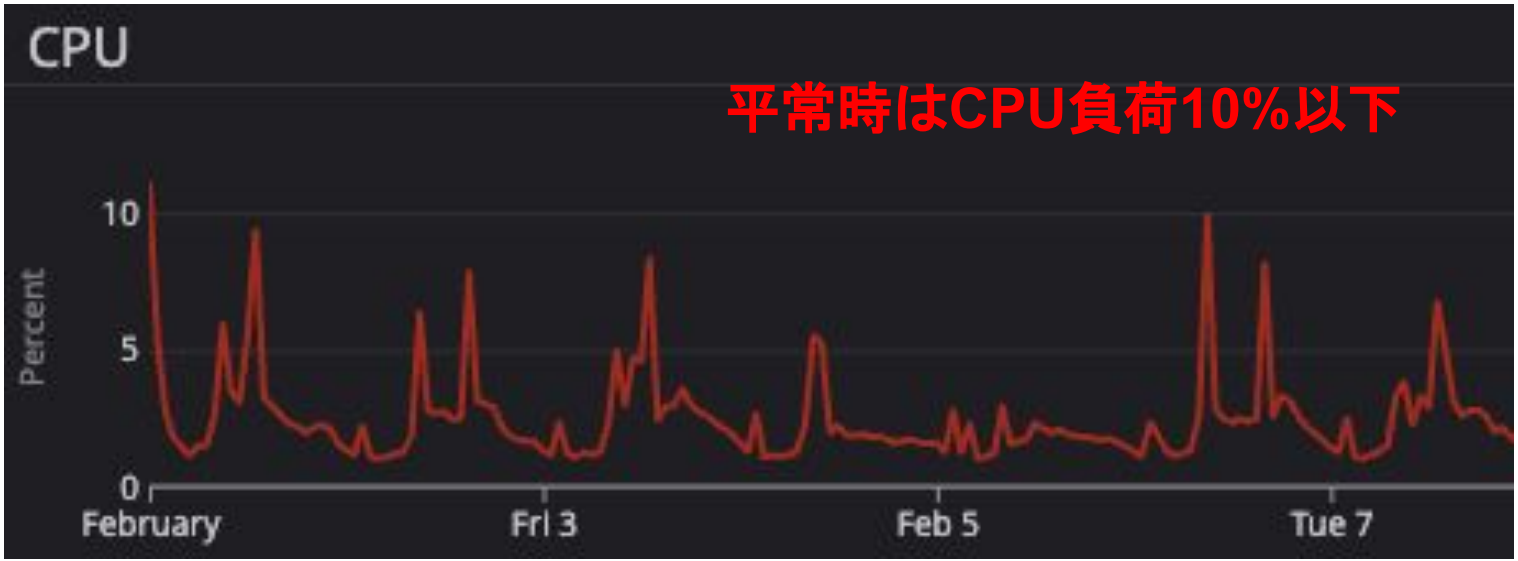
- 5.7から8.0へ
- DATETIME型の比較で明示的にCASTが必要だった
- 上記対応をして5.7系の環境に事前にバックポート

2022/12: RDS for MySQL から Amazon Auroraへ移行

- Aurora MySQL version 3.02.1

Auroraの利用状況

- プロビジョニングされたインスタンス
- インスタンスタイプ: db.r6g.4xlarge



DBもオートスケーリング

できないか？ 🤔

Aurora Serverless v2

要件の確認

- Serverless V1よりもスケーリングの応答性が高く、高可用性
- Aurora MySQL version 3.02.1 のため、移行可能
- 同等性能(ACU換算)では、プロビジョニングされたインスタンスよりも高価
  - アンケートサービスの特性にマッチ

切り替え方法は？

- Amazon Aurora Blue/Green Deployments
- 昨年のRe:Inventで発表
  - 使えるなら使いたい
- 1分程度のダウンタイムで本番DBへ昇格
  - **顧客影響を最小限に**

ということで、

無事移行できました 🎉

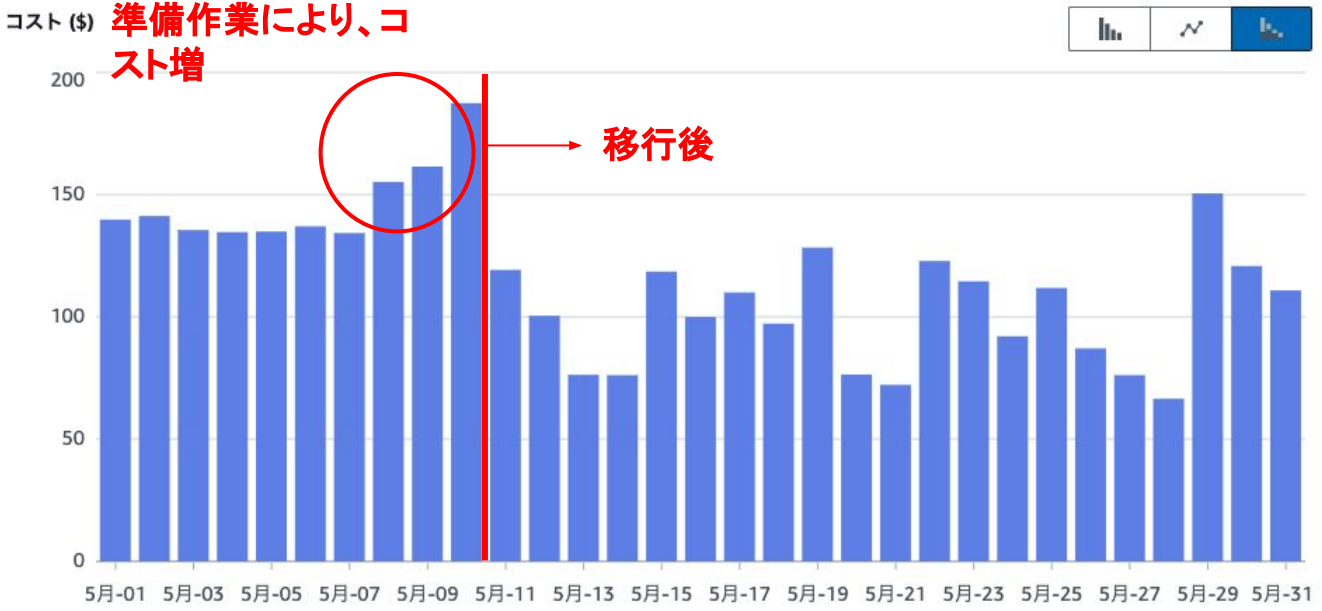
Blue/Greenデプロイメントの事前準備

- サービスの停止を伴うもの
  - binlogの設定を変更
    - `binlog_format`のパラメータ値をMIXEDに変更

- サービスの停止を伴わないもの
  - Blue/Green の構成
    - Green環境を Serverless v2に変更
  - Terraformの微修正
    - 構成管理上の軽微なもの

Blue/Green環境構築コスト

- 一時的にGreen環境のインスタンスを立ち上げる必要がありコストがかかる



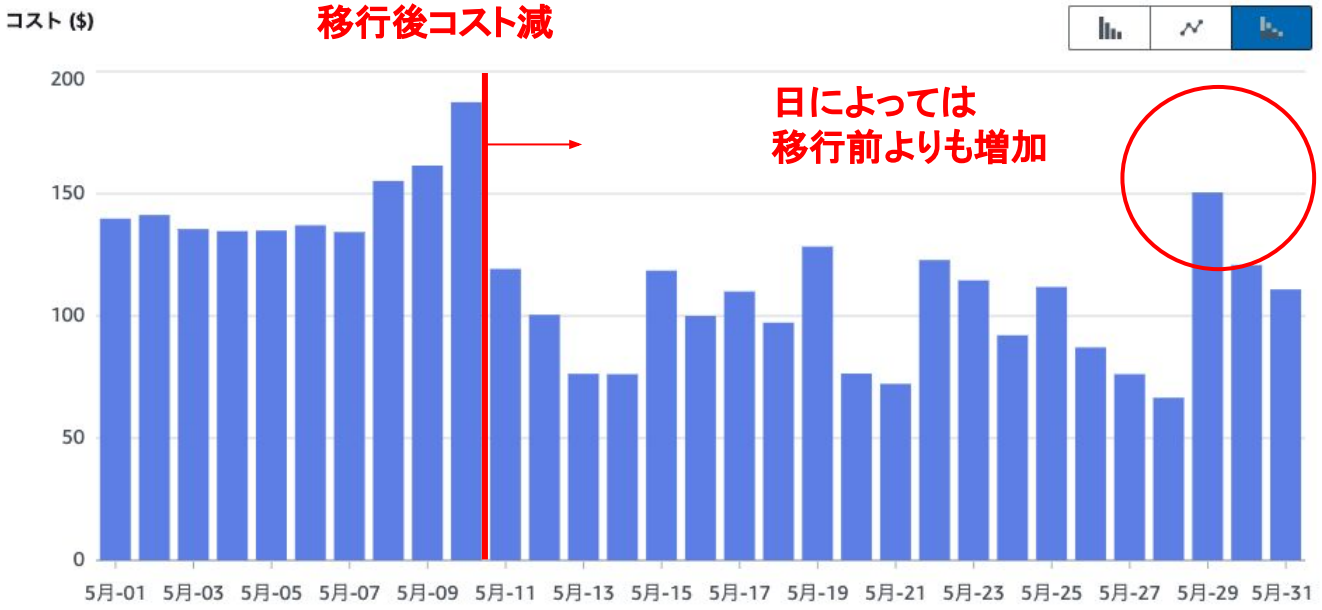
Blue/Greenデプロイメントの実施

- 事前準備を済ませ、AWSコンソール上からスイッチオーバー
  - 1,2分で切り替え完了し、Serverless v2のDBが適用

- 移行時の苦労話は？
  - 特に苦労はなかった
  - Terraform の構成管理についても事前対応のおかげで、運用が変わることはなかった

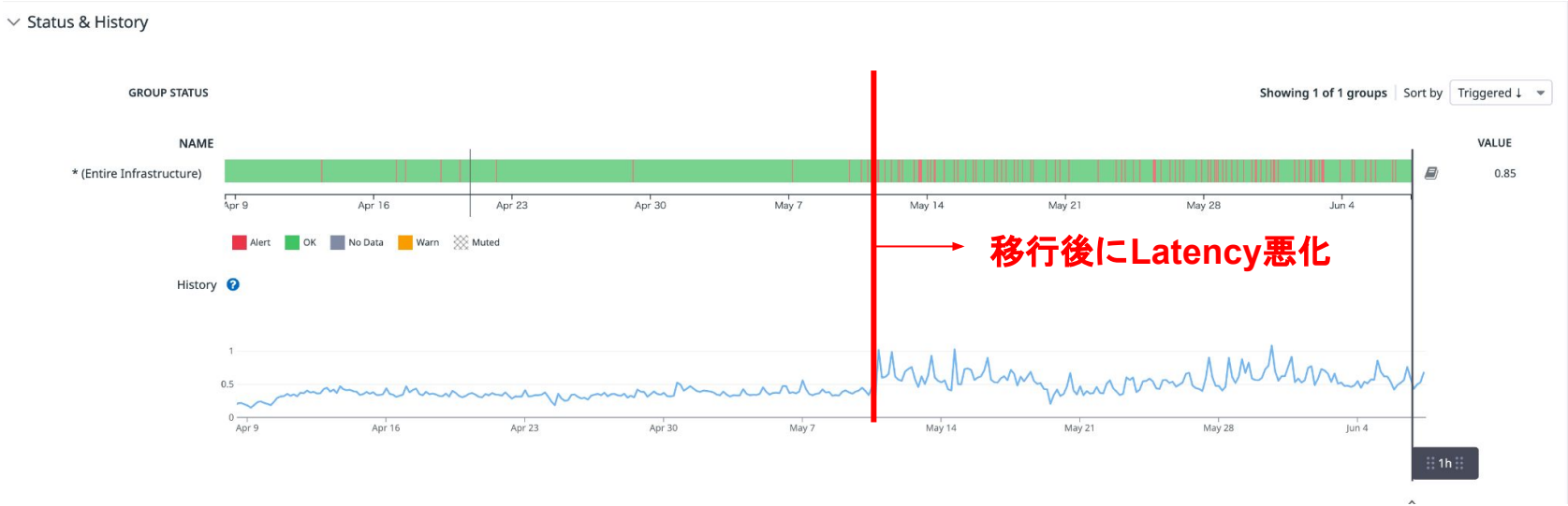
移行後の影響(コスト)

- 1日あたりの平均コスト
  - $136 -> $101 (約25.7%削減)



移行後の影響(パフォーマンス)

- SLIとして設定していた、p90, p99のLatencyが悪化
  - 事前の性能検証で影響範囲ないと想定
  - p99に関してはより顕著な影響



移行後の運用

- ACU(Aurora Capacity Unit)について
  - Serverless v2のキャパシティ単位
  - 1ACU当り、2GiBのメモリに相当
  - 移行前のdb.r6g.4xlarge相当は、64ACU

- ACUの設定
  - Min ACU: 2
  - Max ACU: 64
    - 移行前のインスタンスと同等性能に設定

移行後の問題点

- 移行当初の問題
  - Min: 0.5ACUにしていたため、負荷に追従しきれなかった
  - パフォーマンスインサイトの設定を有効にしているため、推奨値のMin ACU: 2に設定

- 定期的にフェイルオーバーが発生
  - サービス影響は軽微ではある
  - サポートに問い合わせ、AWS側の問題である可能性
  - Auroraのバージョンが若干古いことに起因？

今後について

- データベースアクセスのチューニングが必要
  - 過剰なスペックのDBを利用していたことで問題が隠蔽

- SLIの妥当性検証
  - p99のLatencyについて再考が必要

- フェイルオーバーへの対策
  - Auroraのバージョンを最新化(3.03.1)して様子見
    - Blue/Greenデプロイメントを活用

まとめ 📝

まとめ

- 通常時とピーク時のアクセス特性が大きく異なるアプリケーションにおいてAurora Serverless v2は有用な選択肢

- Blue/Green DeploymentsによるDBの構成変更が容易に

- オートスケールする仕組みだからこそ、DBのパフォーマンスチューニングは不可避