DMSを利用して、オンプレOracleの大規模データを Auroraへ継続的にレプリケーションした事例の紹介

株式会社リクルート
辛剣徳

RECRUIT

目次

自己紹介

辛剣徳(Shin Kento)

2021年 リクルート中途入社

「じゃらんnet」のバックエンド開発・運用を担当

じゃらんnet

じゃらんnetとは

- 宿・ホテル予約のWebサービス
- 宿予約だけでなく、パッケージツアー、レンタカー、ゴルフなどの様々なサービスを提供

360°トラベルパートナー

宿泊施設・地域のパートナーとして、旅行業界に貢献

じゃらんnetのシステム

- 環境：オンプレミス
- アプリケーション：Java
- DB：Oracle

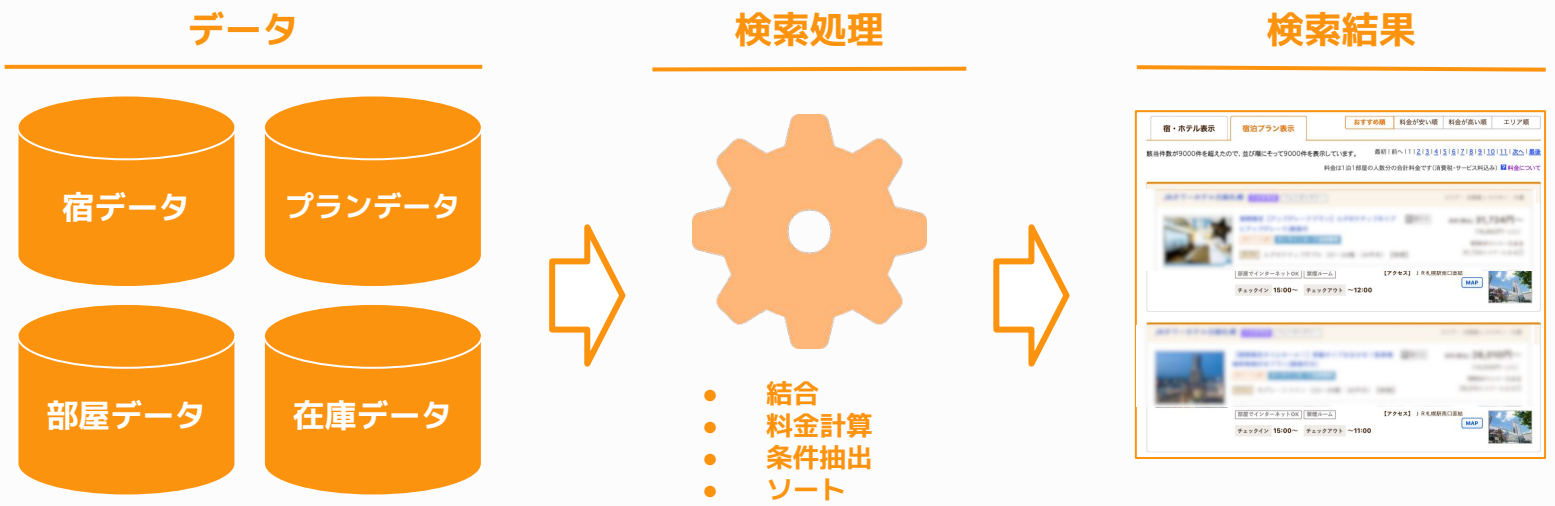
在庫検索機能

じゃらんの在庫検索機能

条件(エリア、日付、予算など)に合う在庫(宿泊施設、部屋タイプ、プランの組み合わせ)を検索する。

在庫検索機能の特徴

- 大量のユーザーからの高頻度なアクセス
- 大規模なデータ量(宿 x プラン x 部屋 x 日付)



アクセス数、処理時間共に大きく、DBに高負荷を与える機能

検索機能の課題

1. DBへの負荷が大きく、オンプレのDBではオンデマンドなスケーリングが難しい
2. 検索以外の機能とも密結合しており、改修時の影響範囲が大きい

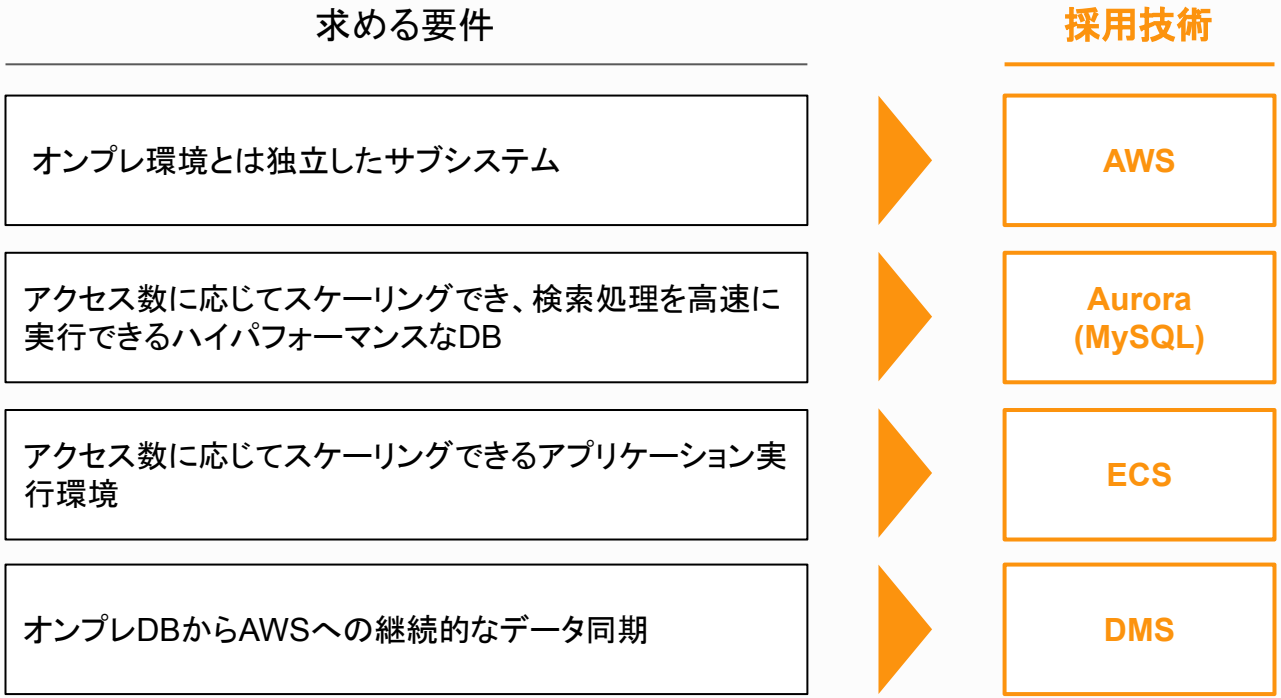
既存のシステムでは、在庫検索機能を利用した新しい機能を追加したくても、コストが大きく、気軽に追加できない

既存環境とは独立した、在庫検索機能を提供するシステムが必要

クラウド在庫検索基盤

クラウド在庫検索基盤とは

在庫検索機能の課題を解決するために構築されたクラウド上の在庫検索API

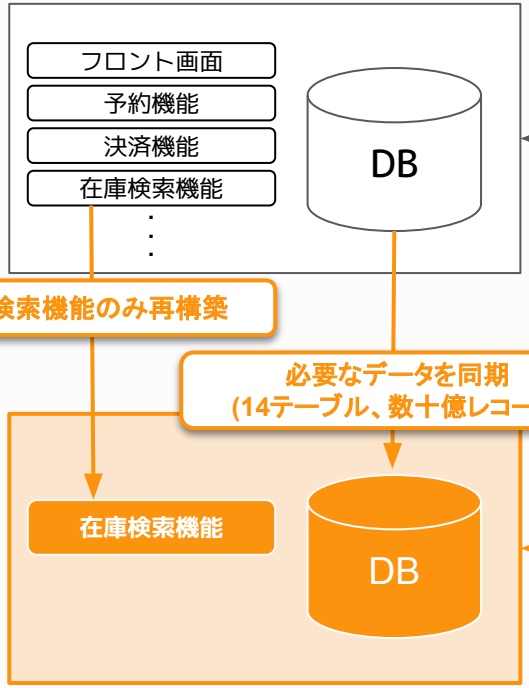


クラウド在庫検索基盤の役割

	本体システム	クラウド在庫検索基盤
役割	検索・予約などの主要機能	UX改善のための新機能や A/Bテスト
環境	オンプレミス	クラウド(AWS)
リクエスト種別	更新リクエストを含む	参照リクエストのみ
データの鮮度	リアルタイム	ほぼリアルタイム (数分程度の遅延は許容)
データの整合性	厳密な整合性が求められる	ある程度の結果整合を許容
セキュリティ	個人情報データを扱う	個人情報データは扱わない

クラウド在庫検索基盤の位置づけ

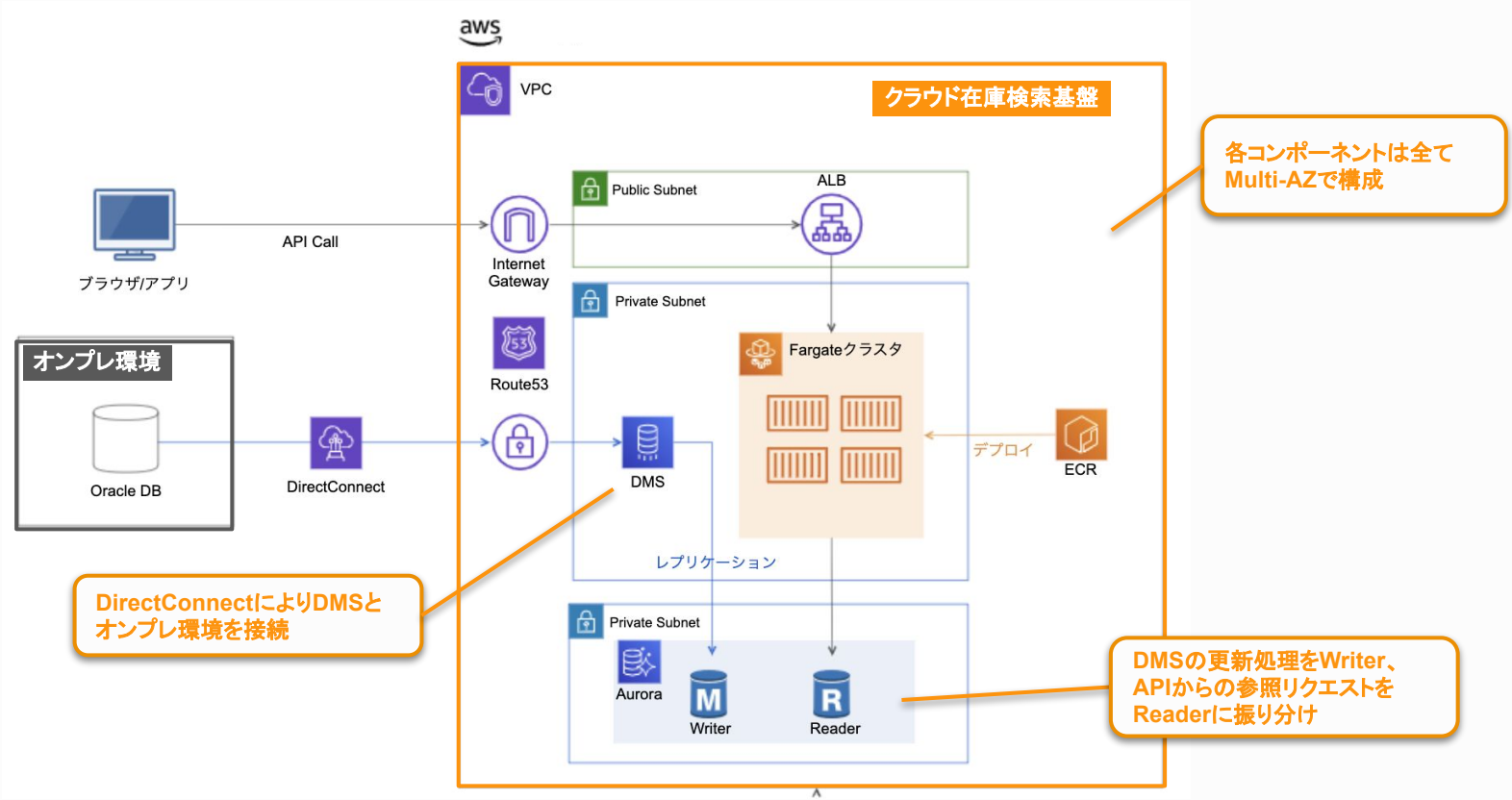
本体システム
(オンプレ)



主要予約導線の検索リクエストは引き
続き本体システムへ

在庫検索機能を利用した新しい機能で
のリクエストはクラウド在庫検索基盤へ

クラウド在庫検索基盤のシステムアーキテクチャ



AWS DMS

AWS DMSとは

- Database Migration Service
- **異種間**のデータベースの移行をサポート
- **ダウンタイムなく**移行が可能
- **CDC(継続的なレプリケーション)**が可能

クラウド在庫検索基盤におけるDMSの利用

DMSの利用により以下を達成

Oracle → MySQLへの異種間DB移行

サービス停止のないオンライン下での全同期

レイテンシ平均1分程度のほぼリアルタイムな同期

DMSの利用手順

① レプリケーションインスタンス、移行タスクを作成

② ターゲットDBにテーブルを準備

③ 既存データの全同期(フルロード)を実施

④ 継続的同期(CDC)を開始

DMSの利用手順① インスタンス、タスクを作成

① レプリケーションインスタンスの作成
インスタンスサイズ等を指定

② エンドポイントの作成
ソース、ターゲットそれぞれに接続するためのエンドポイントを作成

③ タスクを作成
移行タイプにフルロード & CDCを指定

テーブルマッピング

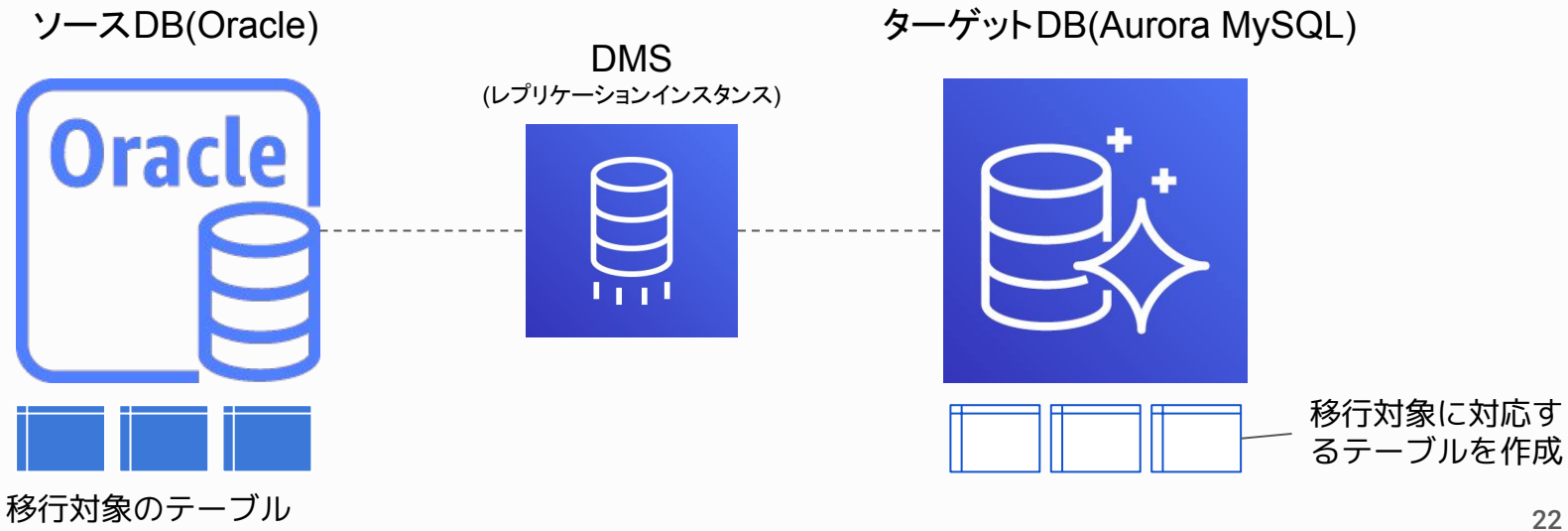
タスク設定のテーブルマッピングを利用することで、連携対象のレコード、カラムを指定することが可能。

- 連携レコードを指定 (選択ルール)

- 連携カラムを指定 (変換ルール)

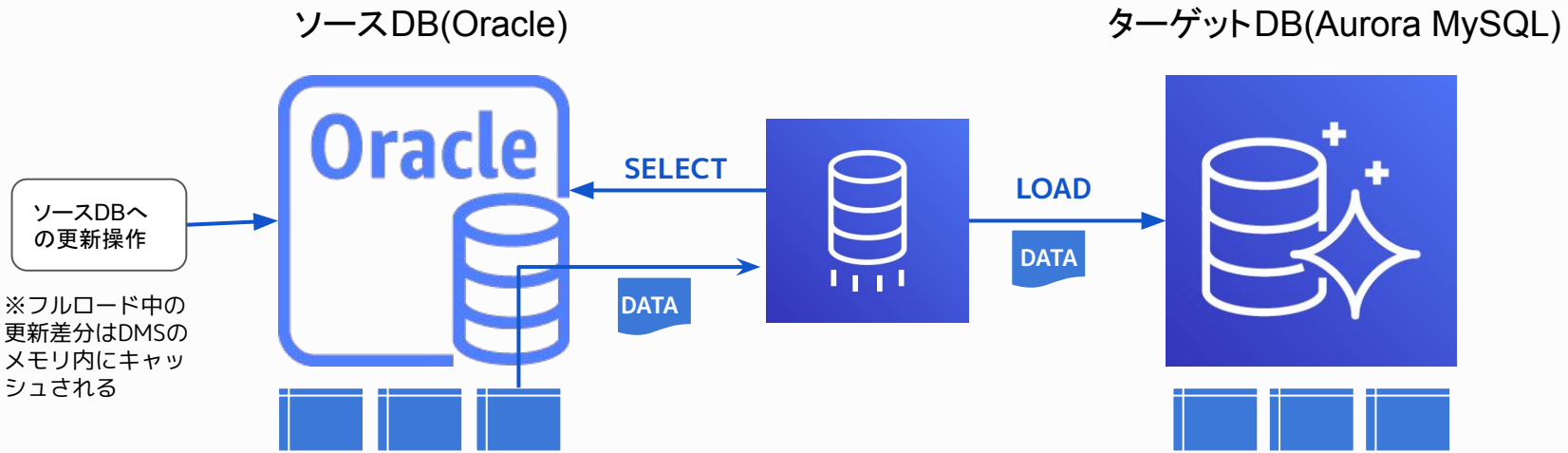
DMSの利用手順② ターゲットテーブルの準備

- 移行対象のテーブルをターゲットDBに作成する。
- Oracleのデータ型に対応する適切なMySQLのデータ型を選択する。
- ロード速度を高めるために、セカンダリインデックスはフルロード完了後に作成する



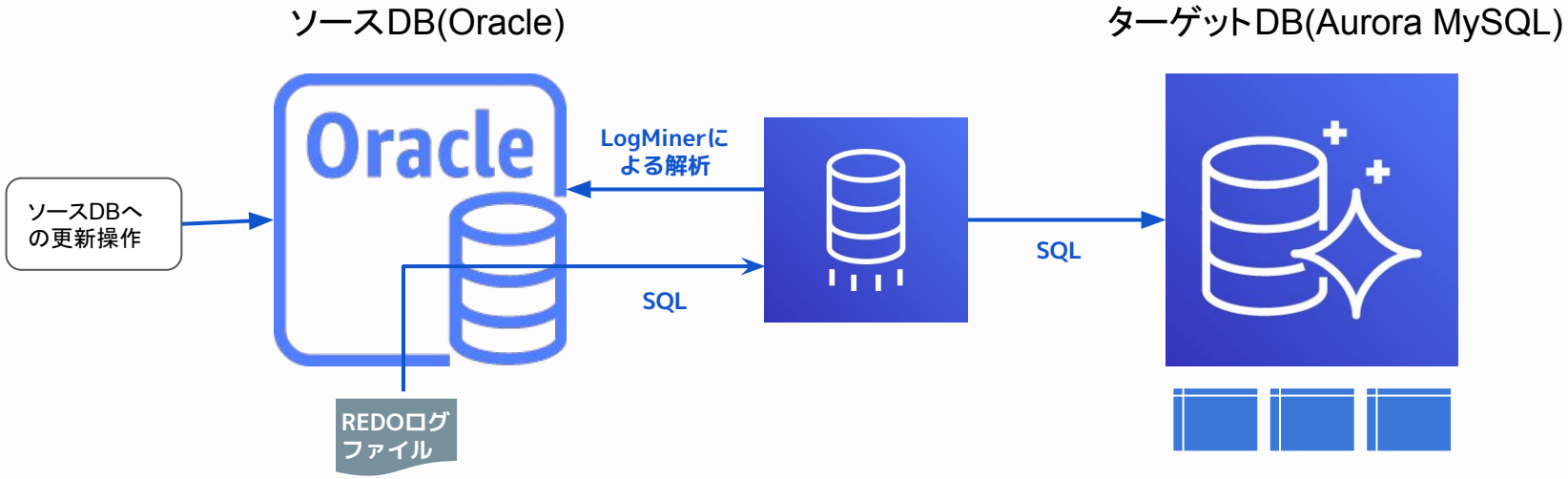
DMSの利用手順②フルロード

- サービス稼働中に既存データの移行 (フルロード)を実施。
- DMSはソースDBにSELECTを実行し、取得したデータをターゲット DBにロードする。
- ソースDBへの負荷を最小限にするため、複数テーブルの並列なロードは行わない。



DMSの利用手順③ CDC

- フルロード中にキャッシュされた変更差分を適用しきった後、新規更新の適用が始まる。
- Oracleがソースの場合、LogMinerによりREDOログファイルを解析し、SQLを生成してターゲットDBに連携する。
  - REDOログファイル: 更新履歴を記録するログファイル
  - LogMiner: Oracleの提供するログファイルの解析機能



フルロード結果

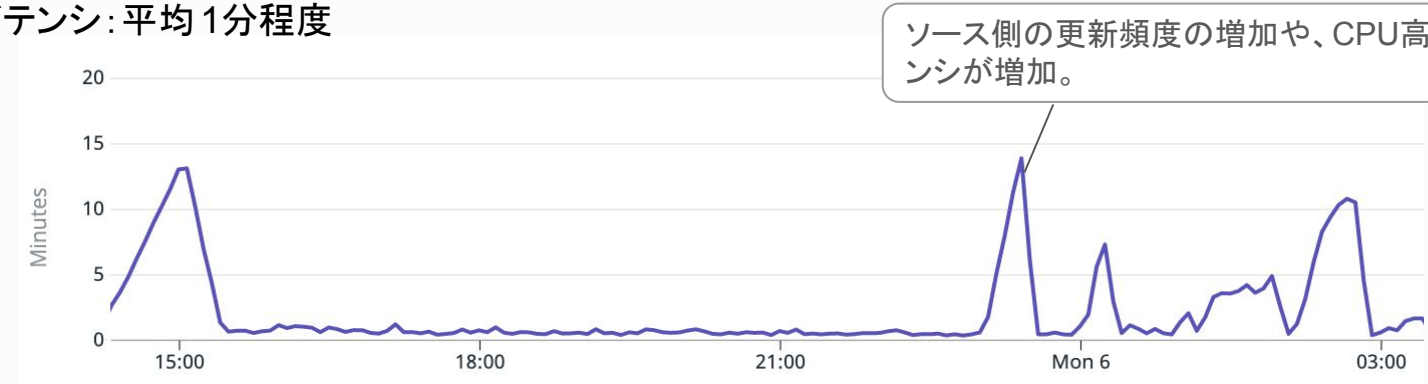
- 合計所要時間 8時間程度
- 平日日中のアクセスの少ない時間帯に、2日に分けて実施。

テーブル	ロード状態	ロード経過時間
Table1	Table completed	< 1 s
Table2	Table completed	< 1 s
Table3	Table completed	3 s
Table4	Table completed	8 s
Table5	Table completed	16 s
Table6	Table completed	33 s
Table7	Table completed	49 s
Table8	Table completed	15 m 38 s
Table9	Table completed	22 m 49 s
Table10	Table completed	41 m 50 s
Table11	Table completed	45 m 27 s
Table12	Table completed	1 h 48 m 5 s
Table13	Table completed	4 h 7 m 31 s

CDC結果

- スループット：平均数 500 rows/s

- レイテンシ：平均 1分程度



DMS利用時の課題と対応

課題① 適切なタスク数の検討

タスクの数だけ並列でCDCを実行できるが、以下のようなトレードオフが存在する

	タスク数を増やした場合	タスク数を減らした場合
レイテンシ	小さい	大きい
DBへの負荷	大きい	小さい
データの整合性	整合性を保ちにくい	整合性を保ちやすい
耐障害性※	高い	低い

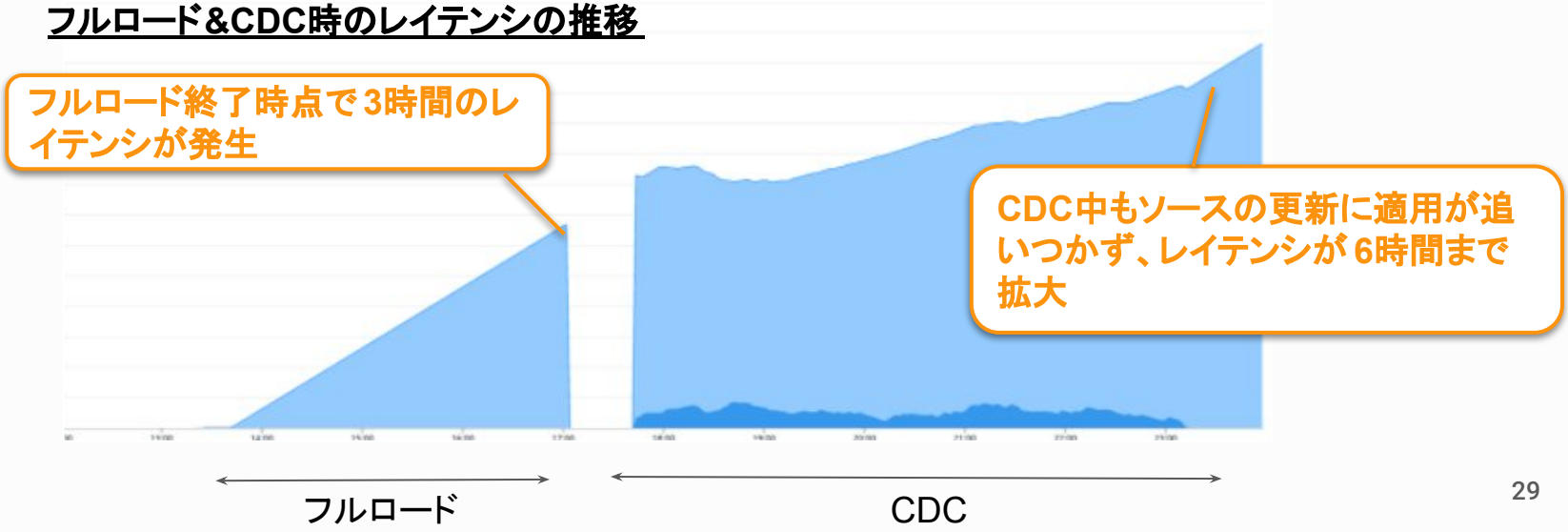
※ タスクが停止した際に影響が及ぶテーブルの数

小規模なテーブルを扱うタスクと大規模なテーブルを扱うタスクの2タスクに分け、大規模テーブルの再ロードリスクを減らす方針

課題② CDCレイテンシの増大

- ソースの更新頻度に対しターゲットへの適用が追いつかず、レイテンシ (同期遅延)が上昇してしまう。
- 特にオンライン下でのフルロードでは、フルロード中の更新差分が蓄積され大きなレイテンシになってしまう。

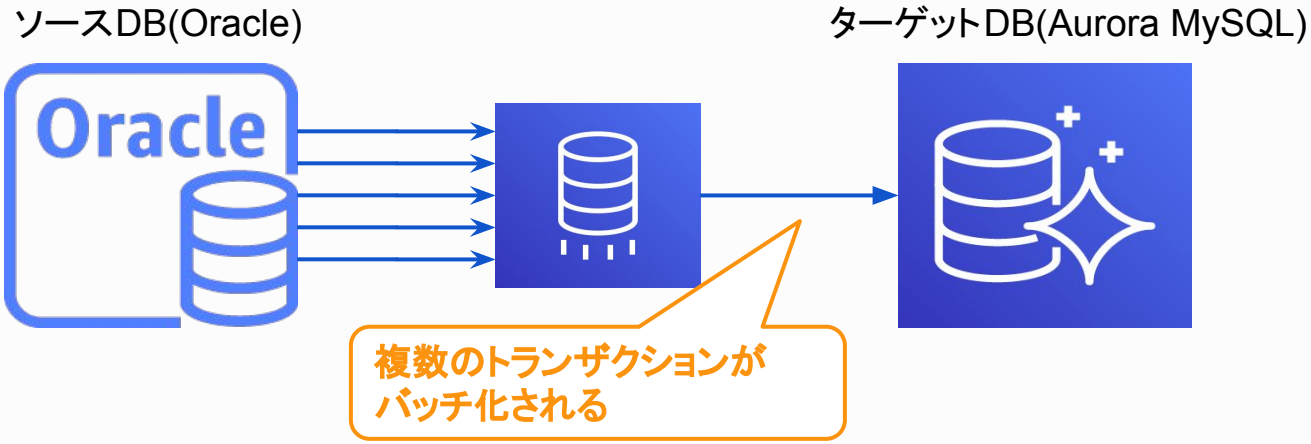
フルロード&CDC時のレイテンシの推移



BatchApply機能

ソースDBで実行された複数のトランザクションを、ひとつの処理にまとめて適用する機能

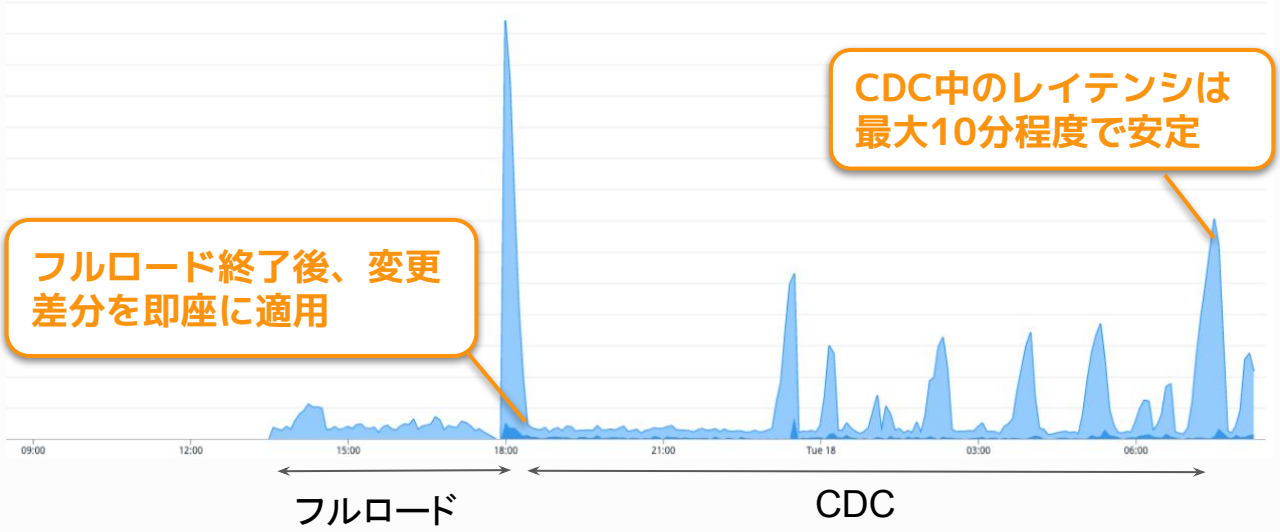
- メリット: ターゲットDBへの適用頻度を抑えることができる
- デメリット: ソースのトランザクション変更されるため、厳密な参照整合性が損なわれる



BatchApply機能の効果

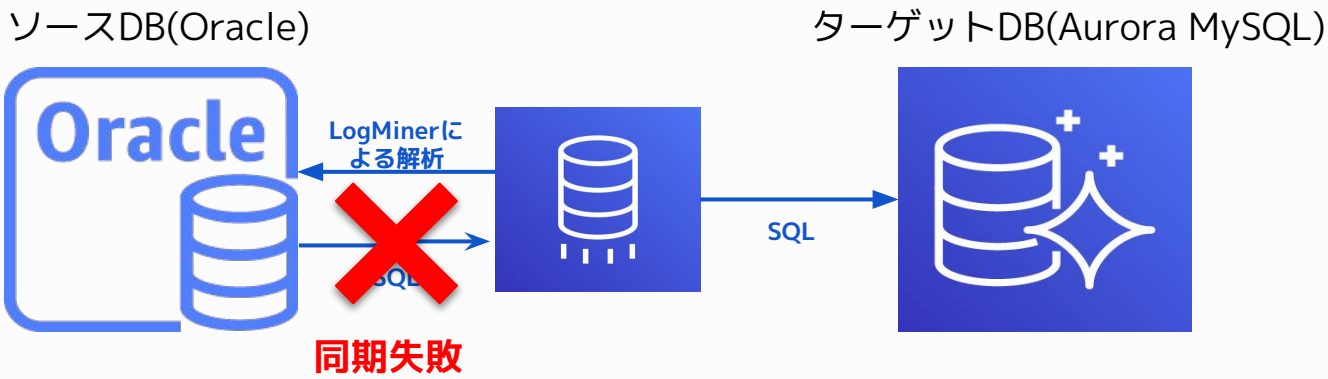
- BatchApplyの有効化により、無効化時と比較してターゲットへの適用スループットが劇的に(100倍程度)改善した。

BatchApply有効化時のレイテンシの推移



課題③ データ差分の発生

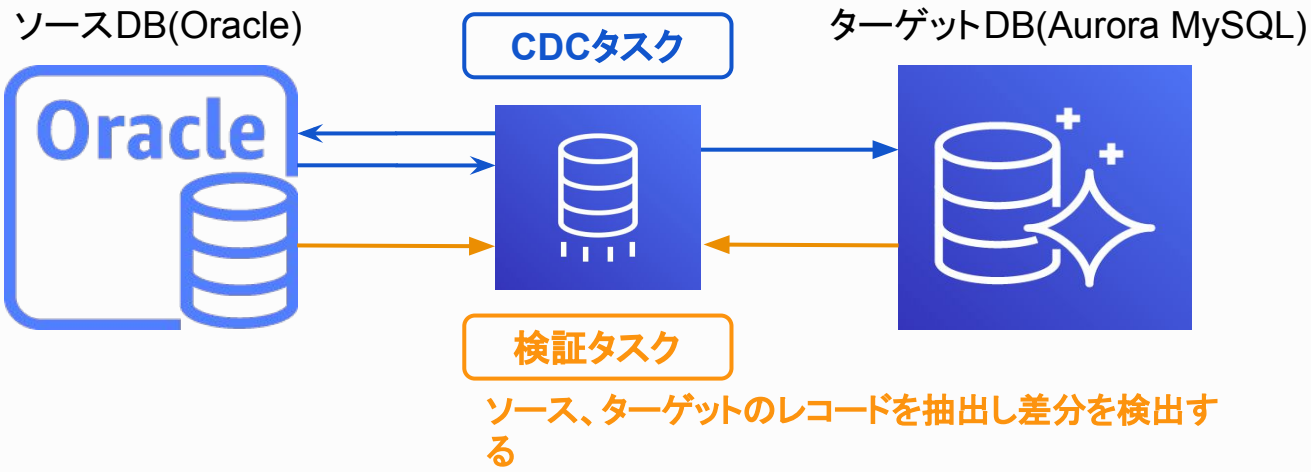
CDCタスクを一時停止後、ソースDBの負荷高騰により再開に一度失敗したことで、一部のINSERT処理を欠損してしまった。(DMSのログから検知)



欠損データの復旧が必要だが、
サービス停止の伴うフルロードはなるべく避けたい

検証タスクの利用

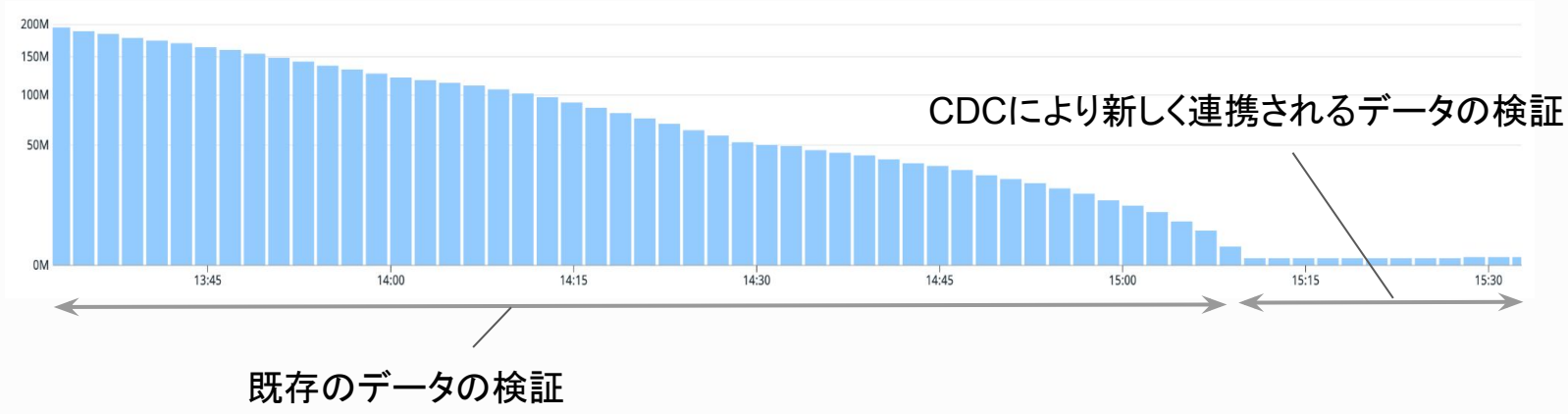
- DMSの**検証タスク**を利用することで、ソース、ターゲットのデータ差分を検出することができる。
- 検証後、フルロードではなく、検知した差分のみを手動で復元することで対応
- 恒常的に検証タスクを稼働させると、ソース DBの負荷が大きくなるため、タスクの停止や不審なログを検知した都度実行する方針を採用。



検証タスク実行結果

- 検証タスクのコンソール画面

- 検証実行時の検証保留中のレコード数の推移



まとめ

DMSを利用しての所感

- DMSにより簡易にデータ移行を実現できるが、機能面、性能面で注意点も多いため、本番適用前に十分に検証してから利用するのが大事。

- タスクのエラーや、データ差分の発生は完全には避けきれないという前提で、失敗した際に再フルロードや検証を実施できる運用にするべき。

導入成果・今後の展望

- 導入成果
  - 平均1分程度のレイテンシでほぼリアルタイムな同期を実現
  - いくつかの新機能やA/Bテストなどで高速に実施できるようになった。

← 在庫検索クラウド基盤利用例
在庫カレンダー画面

- 今後の展望
  - アクセス数の大きい機能や、処理負荷の高い検索処理など本体システムで実現できない機能での利用拡大を目指す。
  - 検索性能そのものを改善し、検索速度、スループットの向上を目指す。

結び

- じゃらんnetでは在庫検索処理の負荷が大きな課題であったが、クラウド在庫検索基盤の導入によって、検索処理の負荷をオフロードし、機能追加のしやすい環境を実現することができた。

- オンプレシステムからクラウドへのデータ同期は、異種DB間の移行・ほぼリアルタイムな同期・大規模データという難易度の高いものだったが、DMSにより実現することができた。

We're Hiring!!!

じゃらんnetでは
バックエンドエンジニア、SREエンジニア等、
様々な職種で一緒に働く仲間を募集しています

- 大規模システムの課題を解決する
- クラウドを武器に開発生産性を上げる

上記に興味がある方はぜひご応募ください！

採用サイト(テクノロジー職): https://recruit-saiyo.jp/technology/
Tech Blog: https://engineers.recruit-jinji.jp/