

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

目次

1. じゃらんnetについて
2. クラウド在庫検索基盤について
3. AWS DMSについて
4. AWS DMS利用時の課題
5. まとめ

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。

条件入力

宿名・キーワードから探す キーワードランキング

例：直前割 東京

日付から探す

宿・ホテル
 JAL 航空券+宿泊
 JR 新幹線・特急+宿泊
 ANA 航空券+宿泊
 遊び・体験(レジャー予約)
 レンタカー

チェックイン 2022/05/05 チェックアウト 2022/05/06

日付未定

目的 出張・ビジネス 旅行・観光

1 部屋 大人 2 名 子供 0名

都道府県選択 エリア選択

禁煙ルーム 温泉 ツイン シングル
 露天風呂付き客室 ハイクラス

検索履歴から探す

日付未定 1部屋 大人2名 札幌

[他の検索履歴を見る](#)

検索結果

宿・ホテル表示 宿泊プラン表示

該当件数が9000件を超えたので、並び順によって9000件を表示しています。 最初 | 前へ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 次へ | 最後

料金は1泊1部屋の人数分の合計料金です(消費税・サービス料込み)

エリア：北海道 > ススキノ・大通

宿泊施設

プラン

部屋タイプ

期間限定【アップグレードプラン】エグゼクティブタイプ 11 時のみ 合計(税込) 31,724円～
(15,862円～)

ポイントUP オンラインカード決済専用

ダブル エグゼクティブダブル (31～34階：24平米) 【禁煙】 31,724スコア～たまる

部屋でインターネットOK 禁煙ルーム 【アクセス】 JR札幌駅南口徒歩

チェックイン 15:00～ チェックアウト ～12:00

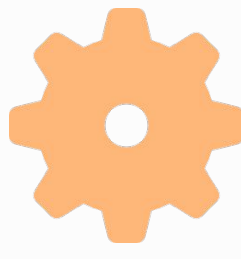
在庫検索機能の特徴

- 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)

データ



検索処理



- 結合
- 料金計算
- 条件抽出
- ソート

検索結果



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

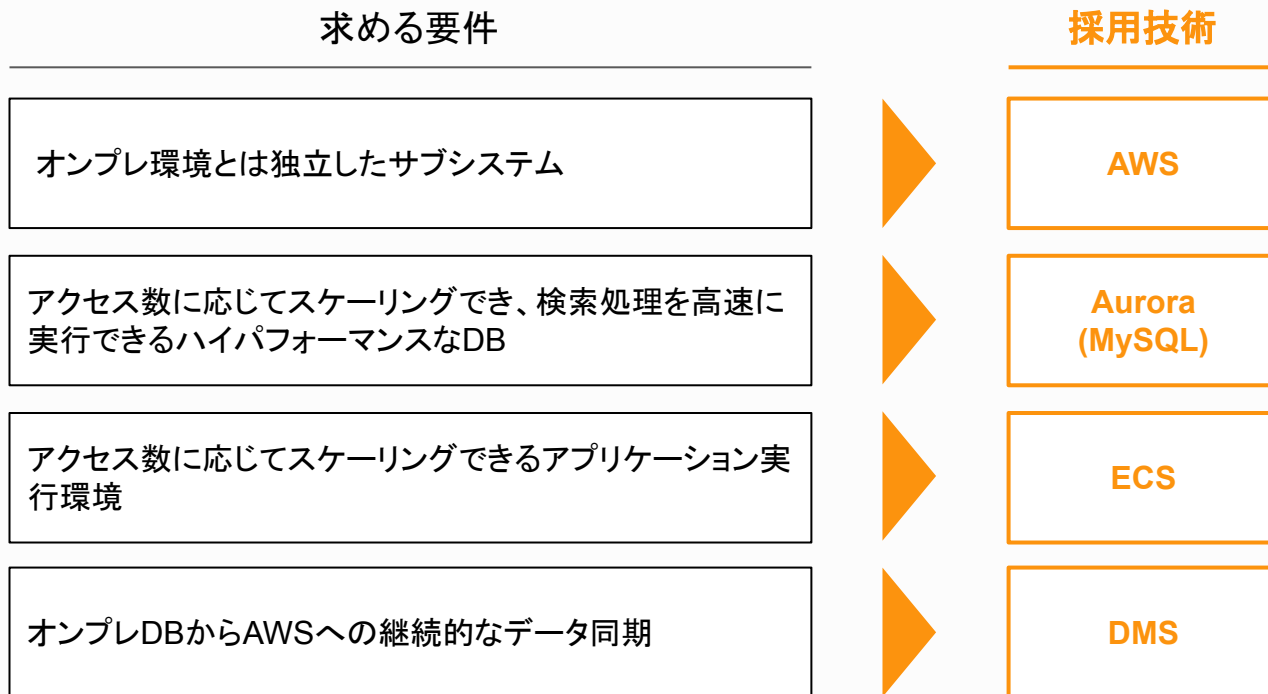


既存環境とは独立した、在庫検索機能を提供するシステムが必要

クラウド在庫検索基盤

クラウド在庫検索基盤とは

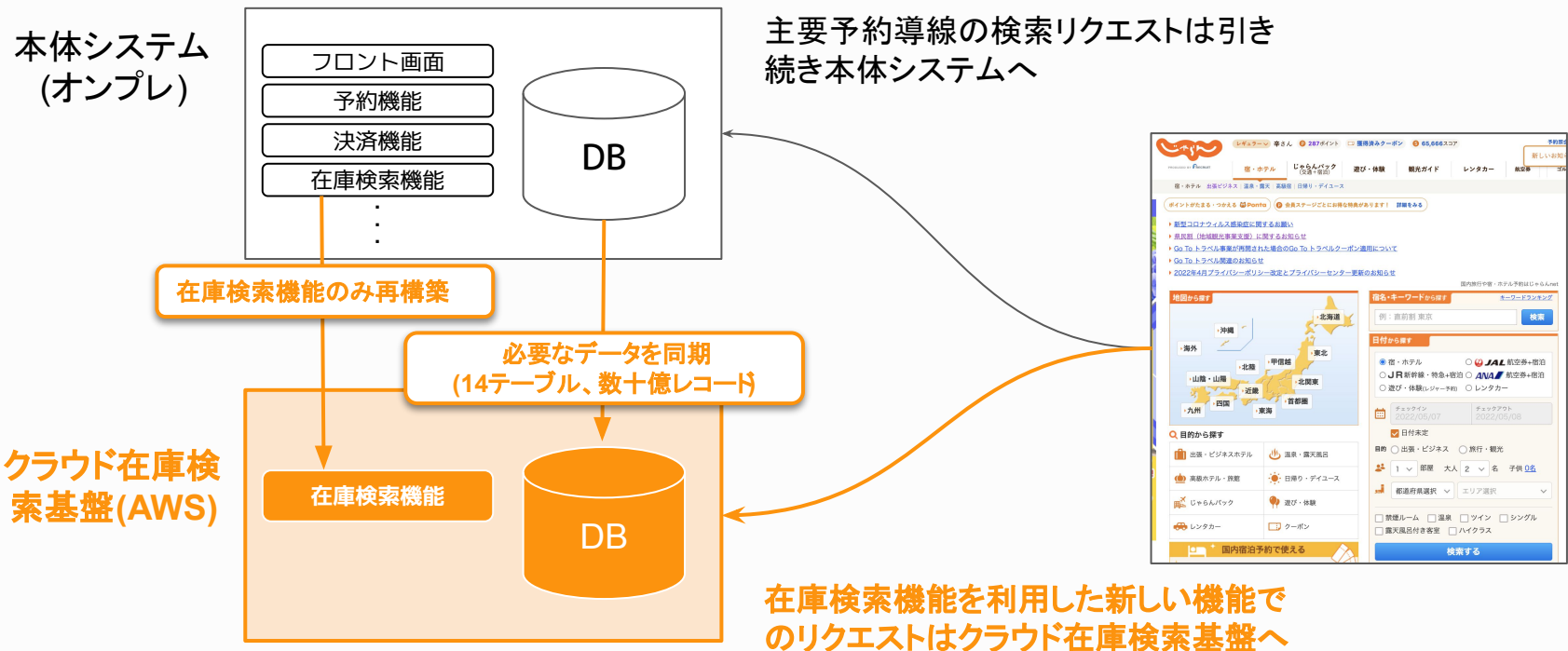
在庫検索機能の課題を解決するために構築されたクラウド上の在庫検索API



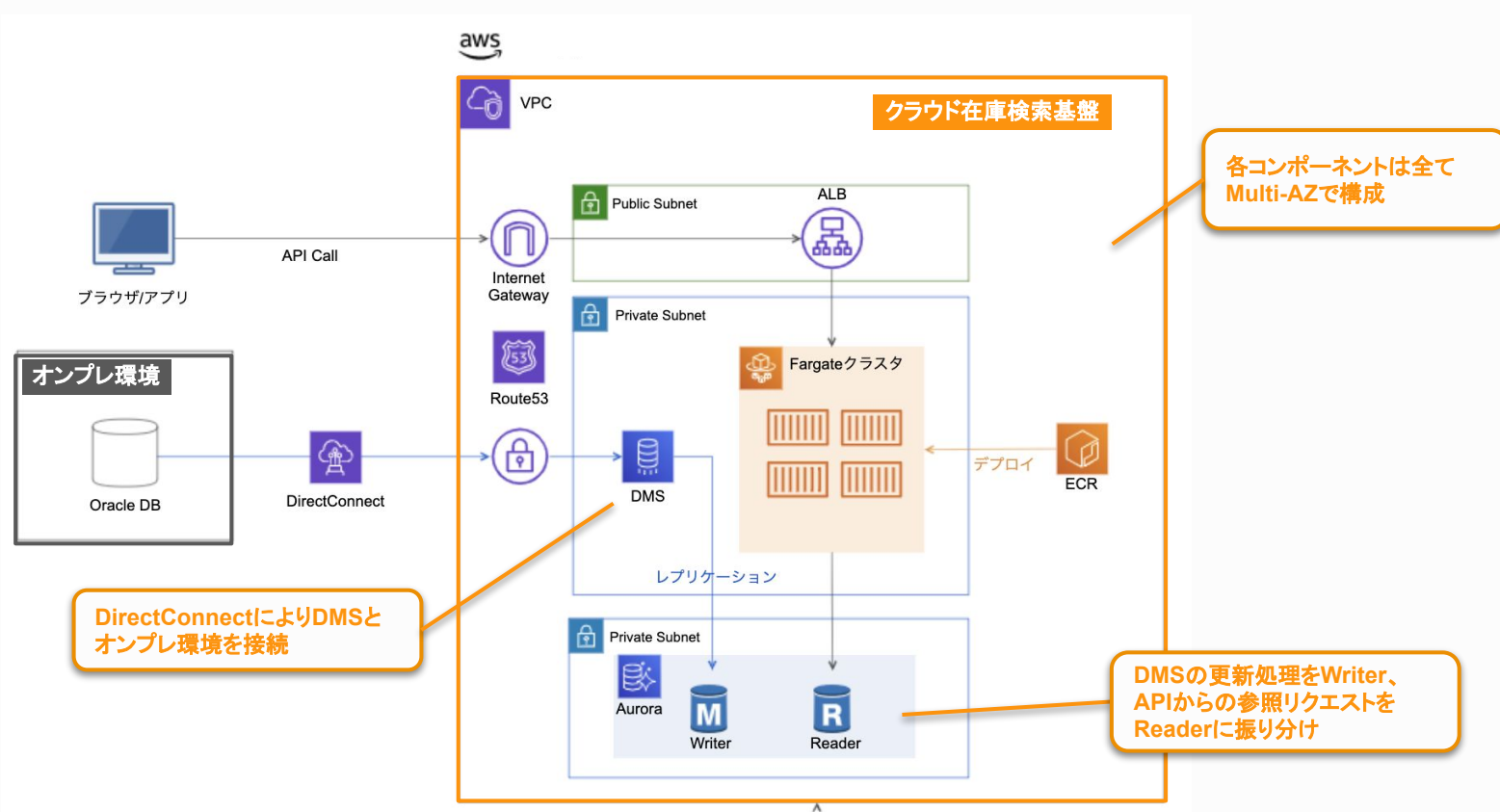
クラウド在庫検索基盤の役割

	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



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

レプリケーションインスタンスの作成

レプリケーションインスタンスの設定

名前
この名前は、現在の AWS リージョン内のすべてのレプリケーションインスタンスで一意である必要があります。

レプリケーションインスタンスの一意の名前を入力します

レプリケーションインスタンス名は数値で始めることはできません

説明的な Amazon リソースネーム (ARN) - 任意
デフォルトの DMS ARN を上書きするためのフレンドリ名です。作成後に変更することはできません。

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

エンドポイント (2)				
<input type="checkbox"/>	名前	タイプ	ステータス	エンジン
<input type="checkbox"/>	dev1-dms-source-endpoint	ソース	アクティブ	Oracle
<input type="checkbox"/>	dev1-dms-target-endpoint	ターゲット	アクティブ	Amazon Aurora MySQL

③ タスクを作成 移行タイプにフルロード & CDCを指定

データ移行タスクの作成

タスクの設定

タスク識別子
タスクの一意の識別子を入力

説明的な Amazon リソースネーム (ARN) - 任意
デフォルトの DMS ARN を上書きするためのフレンドリ名です。作成後に変更することはできません。

Friendly-ARN-name

レプリケーションインスタンス
レプリケーションインスタンスを選択

ソースデータベースエンドポイント
dev1-dms-source-endpoint

ターゲットデータベースエンドポイント
dev1-dms-target-endpoint

移行タイプ 情報
既存のデータを移行して、継続的な変更をレプリケートする

タスク設定

編集モード 情報

ウィザード
使用可能なタスク設定のリセットのみを入力できます。

JSON エディタ
使用可能なすべてのタスク設定を JSON 形式で直接入力できます。

ターゲットテーブル作成モード 情報

何もしない

ターゲット上のテーブルを削除

TRUNCATE

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)

▼ 選択ルール

移行タスクに含めるか、移行タスクから除外するスキーマとテーブルを選択します。 [新しい選択ルールの追加](#)

▼ 場所 スキーマ名 類似 * および テーブル名 類似 %', include

スキーマ
スキーマの選択

テーブル名
ワイルドカードとして % 文字を使用する
%

アクション
選択したオブジェクトを移行するには「Include」
含む

ソースフィルタ [情報](#)

▼ 列フィルター 1

列名
DATE

条件 1
以上 2022-06-01

条件の追加

特定の日付以降の在庫データに制限

- 連携カラムを指定 (変換ルール)

▼ 変換ルール

変換ルールを使用して、選択した一部またはすべてのオブジェクトのスキーマ、テーブル、または列名を変更または変換できます。 [情報](#) [変換ルールの追加](#)

▼ 場所 スキーマ名 類似 %' および テーブル名 類似 %', remove-column

ターゲット
列

スキーマ名
スキーマの選択

テーブル名
ワイルドカードとして % 文字を使用
%

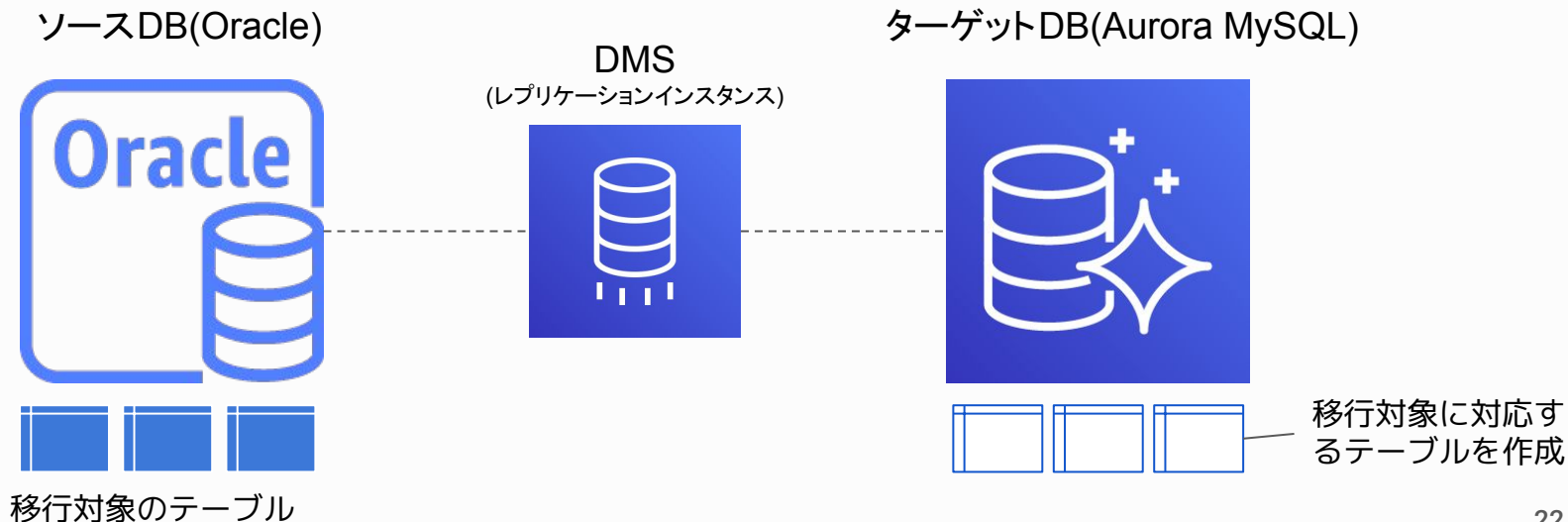
列名
SECRET_COLUMN

アクション
列の削除

個人情報カラムを連携対象から除外

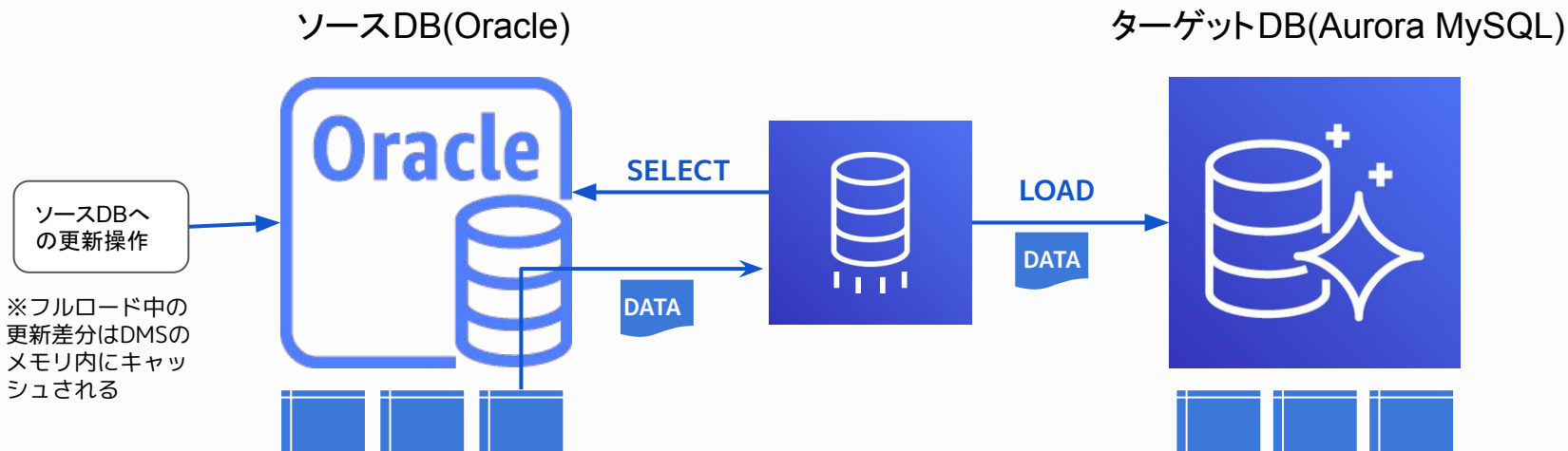
DMSの利用手順② ターゲットテーブルの準備

- 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



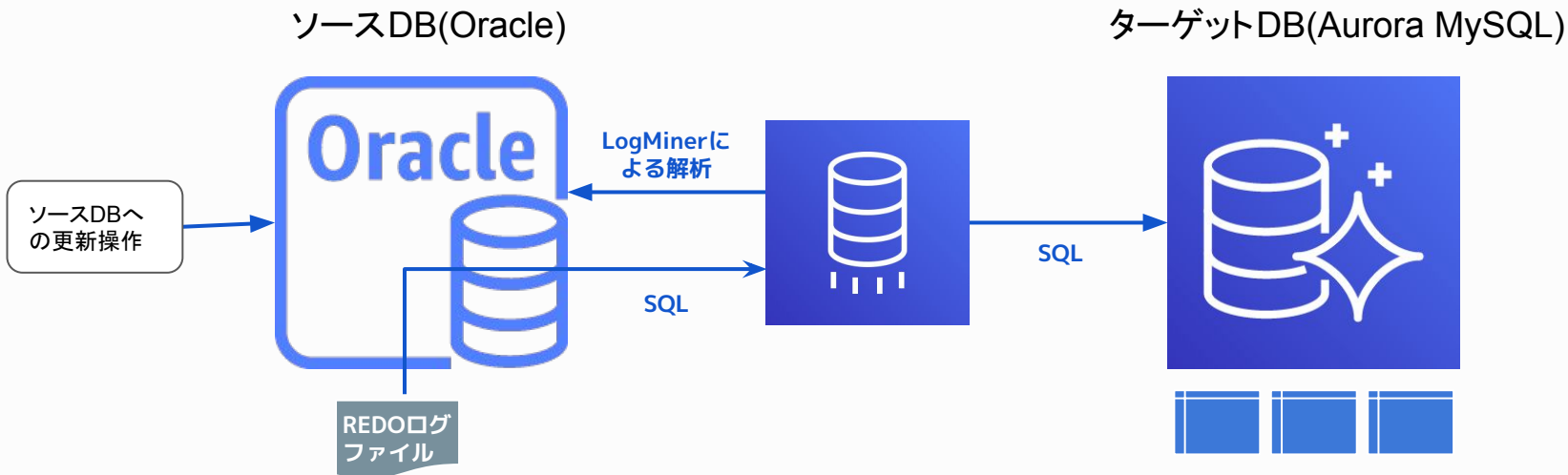
DMSの利用手順②フルロード

- 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



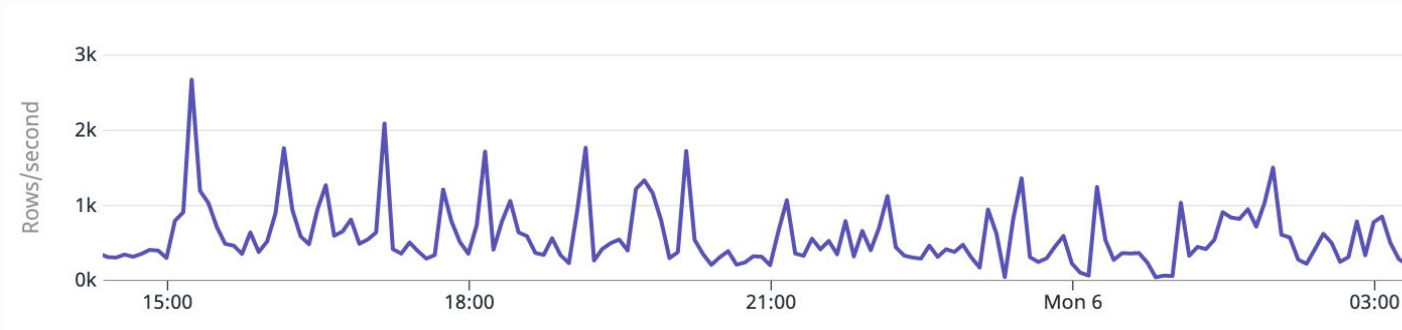
フルロード結果

- 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

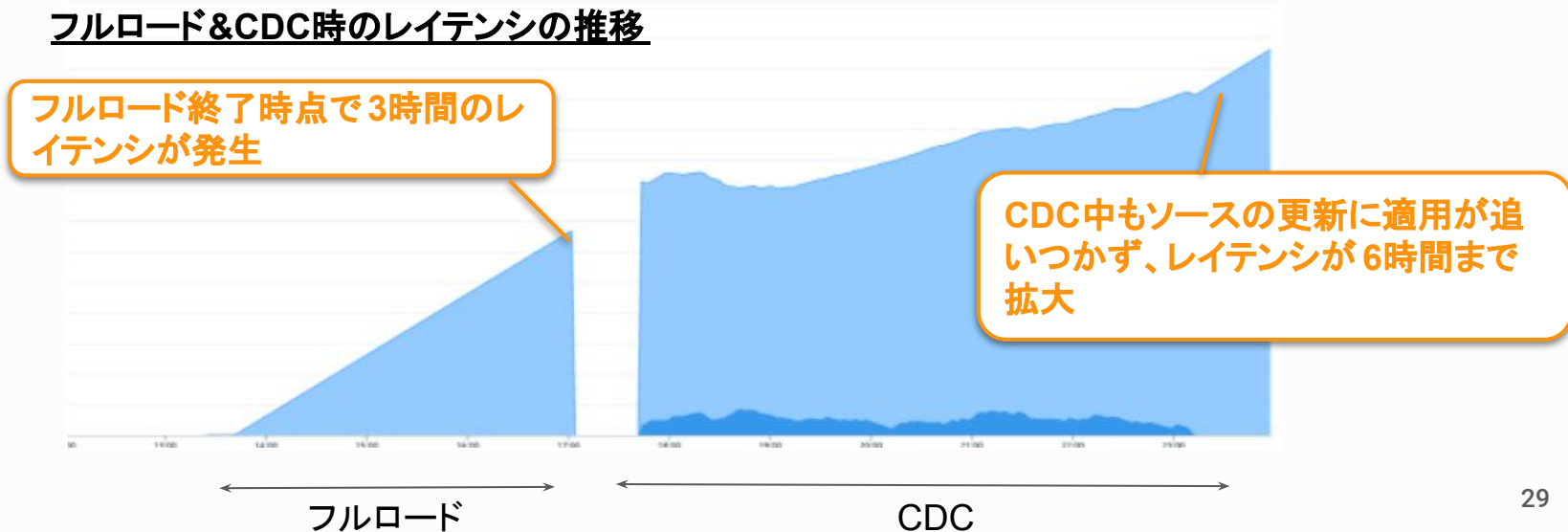


小規模なテーブルを扱うタスクと大規模なテーブルを扱うタスクの2タスクに分け、大規模テーブルの再ロードリスクを減らす方針

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。

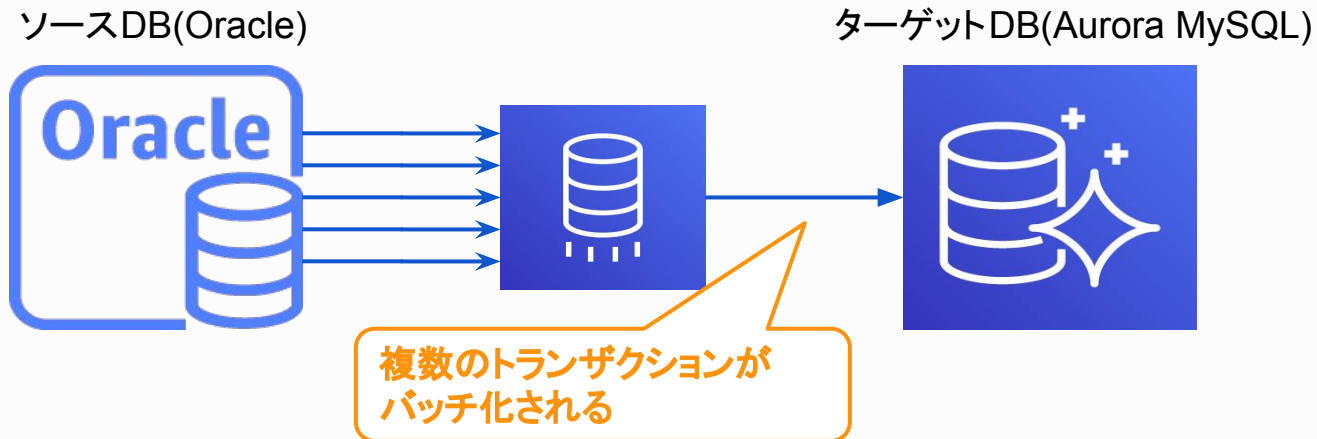
フルロード&CDC時のレイテンシの推移



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

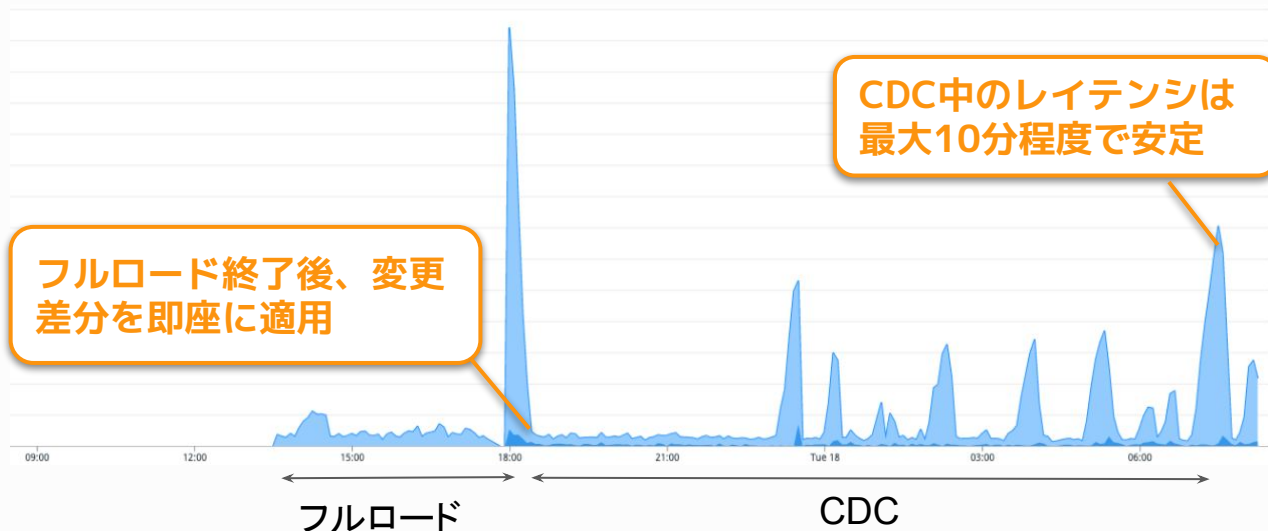
- メリット: ターゲットDBへの適用頻度を抑えることができる
- デメリット: ソースのトランザクション変更されるため、厳密な参照整合性が損なわれる



BatchApply機能の効果

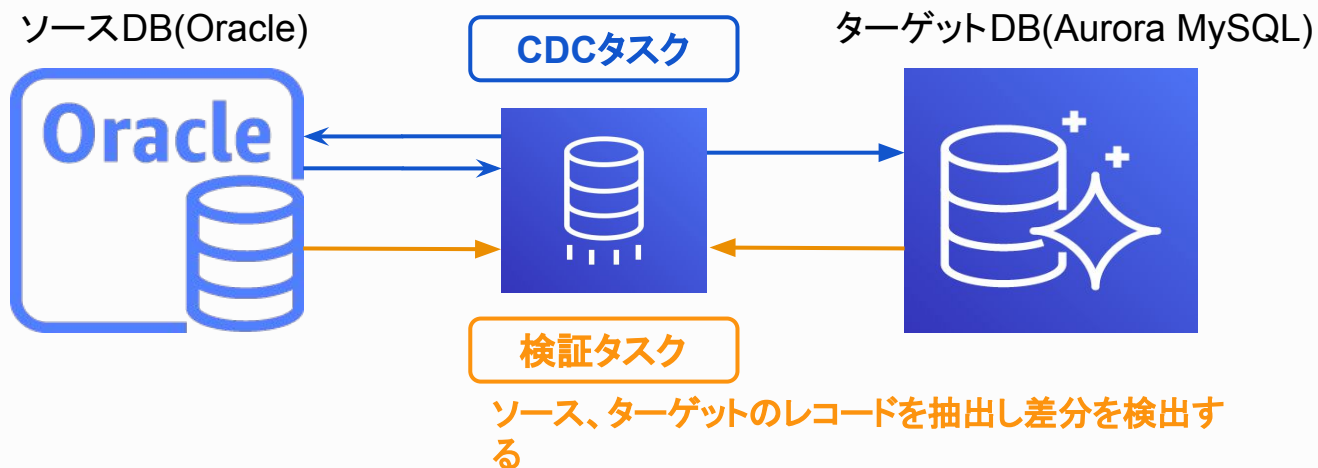
- BatchApplyの有効化により、無効化時と比較してターゲットへの適用スループットが劇的に(100倍程度)改善した。

BatchApply有効化時のレイテンシの推移



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。

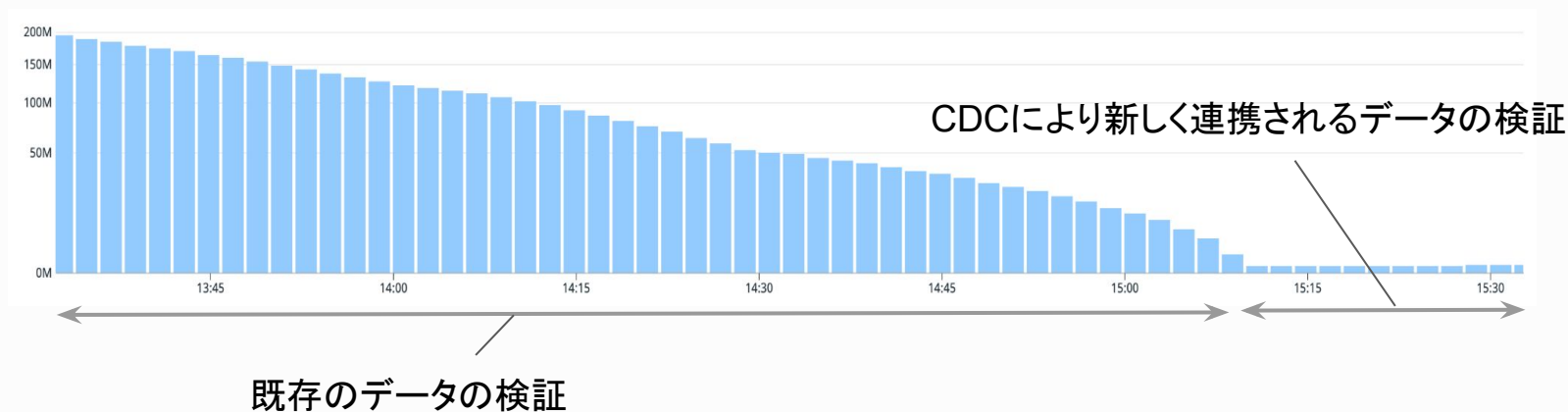


検証タスク実行結果

- 検証タスクのコンソール画面

Applied deletes	Applied updates	Applied DDLs	行のフルロード	行の合計	検証状態	検証保留中
0	0	0	0	0	Pending records	84,888

- 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。

The screenshot shows a web interface for searching inventory. At the top, there are tabs for '施設情報' (Facility Info), '宿泊プラン (295件)' (Stay Plans (295 items)), '写真' (Photos), 'クチコミ' (Reviews), and '地図・アクセス' (Map/Access). Below the tabs are filters for '日付未定' (Date unspecified) and '1部屋 大人2名' (1 room, 2 adults). There is a button to 'こだわり条件を追加する' (Add custom conditions). Below that is a '空室情報' (Vacancy Info) section with a link to '空室情報をもっとみる' (View more vacancy info). At the bottom is a calendar grid for the month of May, with dates from 5/26 to 6/1. The 26th is marked with an 'x', and the 28th, 29th, 30th, 31st, and 1st are marked with blue circles.

← 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、
様々な職種と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/>