Amazon Redshift Integration Deep Dive

アマゾン データ サービス ジャパン 株式会社
八木橋 徹平

amazon web services

AWS Summit

Tokyo

amazon web services

■Gold Sponsors

NTT DATA

Serverworks

Empowered by Innovation
NEC

NetApp

HITACHI
Inspire the Next

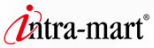
■Global Sponsors

■Silver Sponsors

JBCC

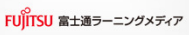
SKYARCH
あなたの前で、あなた以上に考える。

■Bronze Sponsors



■Global Tech Sponsors

■Logo Sponsors



ハッシュタグ **#AWSSummit** で、**皆さんのツイート**が展示エリアの**大画面に表示されます**

公式アカウント**@awscloud_jp** を**フォローすると**、ロゴ入りコースターを**プレゼント**

【コースター配布場所】
メイン展示会場、メイン会場1F受付、デベロッパーカンファレンス会場

amazon web services

自己紹介

- 名前
  - 八木橋 徹平（やぎはし てっぺい）
- 所属
  - アマゾンデータサービスジャパン株式会社
    技術統括本部ストラテジックソリューション部
    ソリューションアーキテクト
- 好きなAWS サービス
  - Amazon Redshift、Amazon Kinesis
  - AWS SDK（Java、Node.js）

amazon web services

セッションの目的

Amazon Redshiftのメリットや使い方を理解されている方に、自社システムや他のAWSサービスとの具体的（且つ低コスト）な連携方法の知識を深めていただく

amazon web services

アジェンダ

- Redshiftの概要
- Redshift主要アップデート
- Redshiftにおけるインテグレーションとは？
- ETL(**E**xtract **T**ransform **L**oad)＋**U**pload
- まとめ

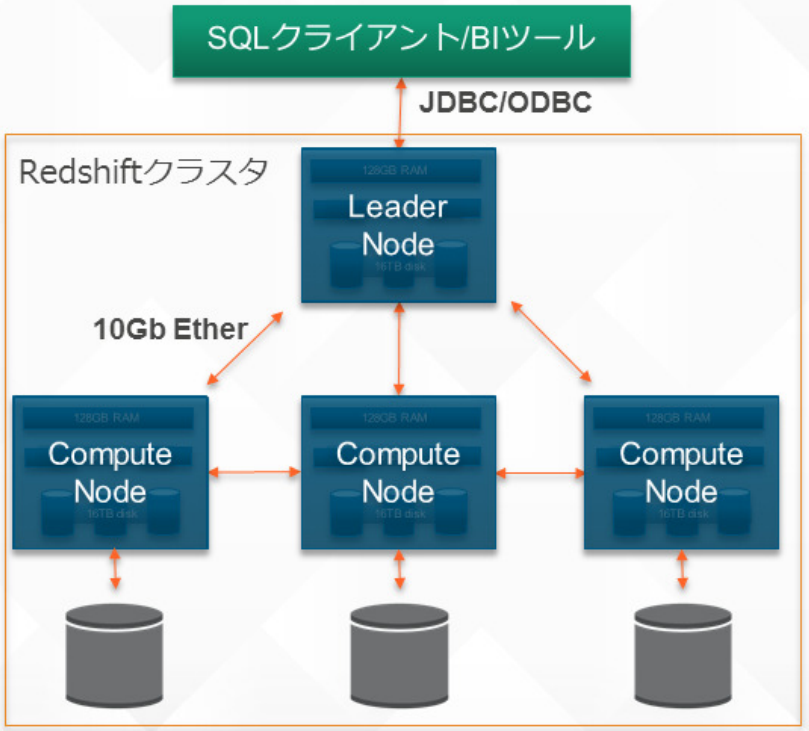
amazon web services

Redshiftの概要

amazon web services

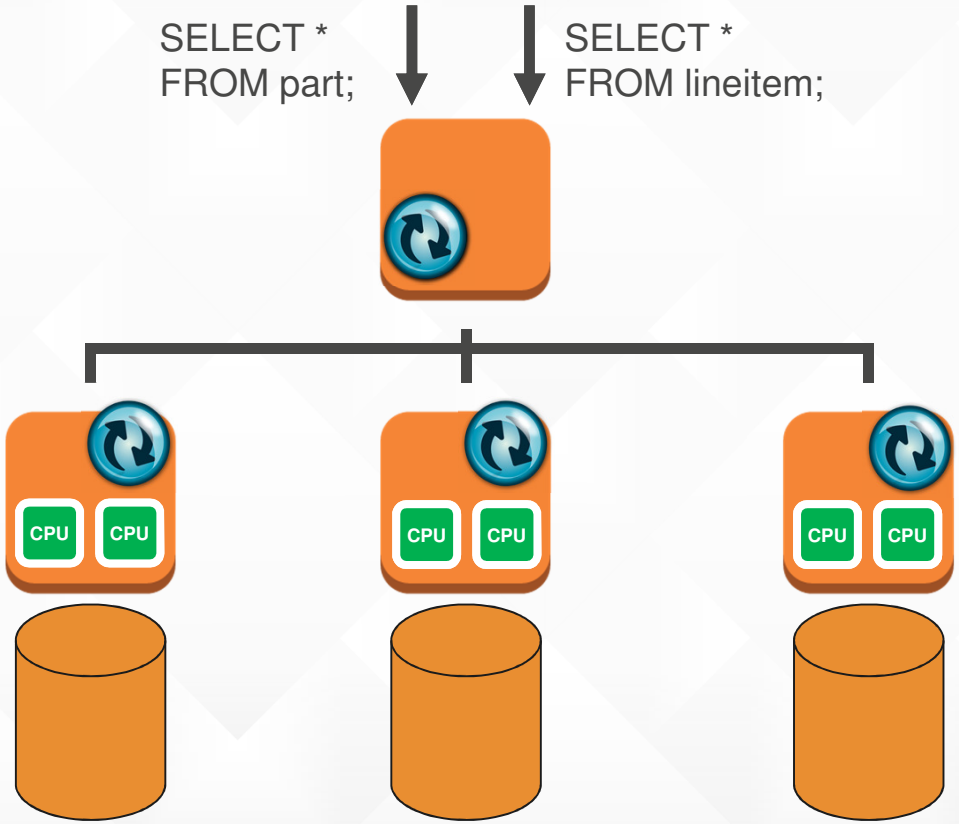
Redshiftのアーキテクチャ

- MPP（超並列演算）
  - CPU、Disk・Network I/Oの並列化
  - 論理的なリソースの括り「ノードスライス」
- データの格納
  - 列指向（カラムナ）
  - 圧縮
- 他のAWSサービスとの親和性



amazon web services

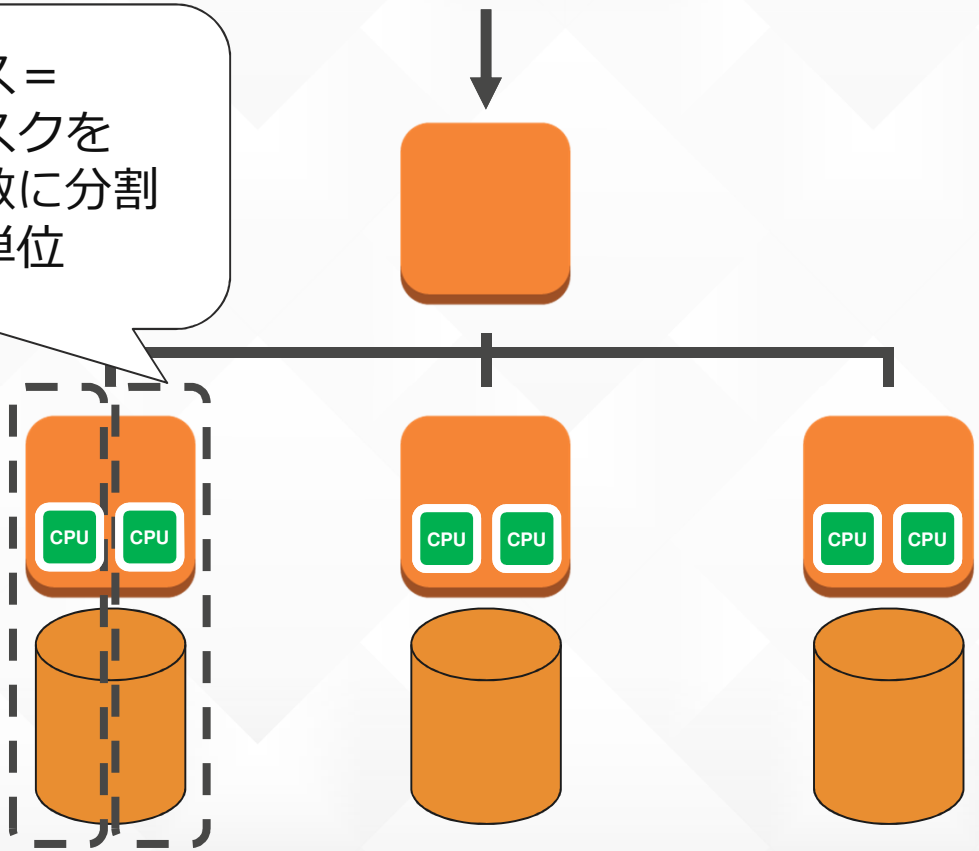
アーキテクチャ：クエリーの並列実行



amazon web services

アーキテクチャ：ノードスライス

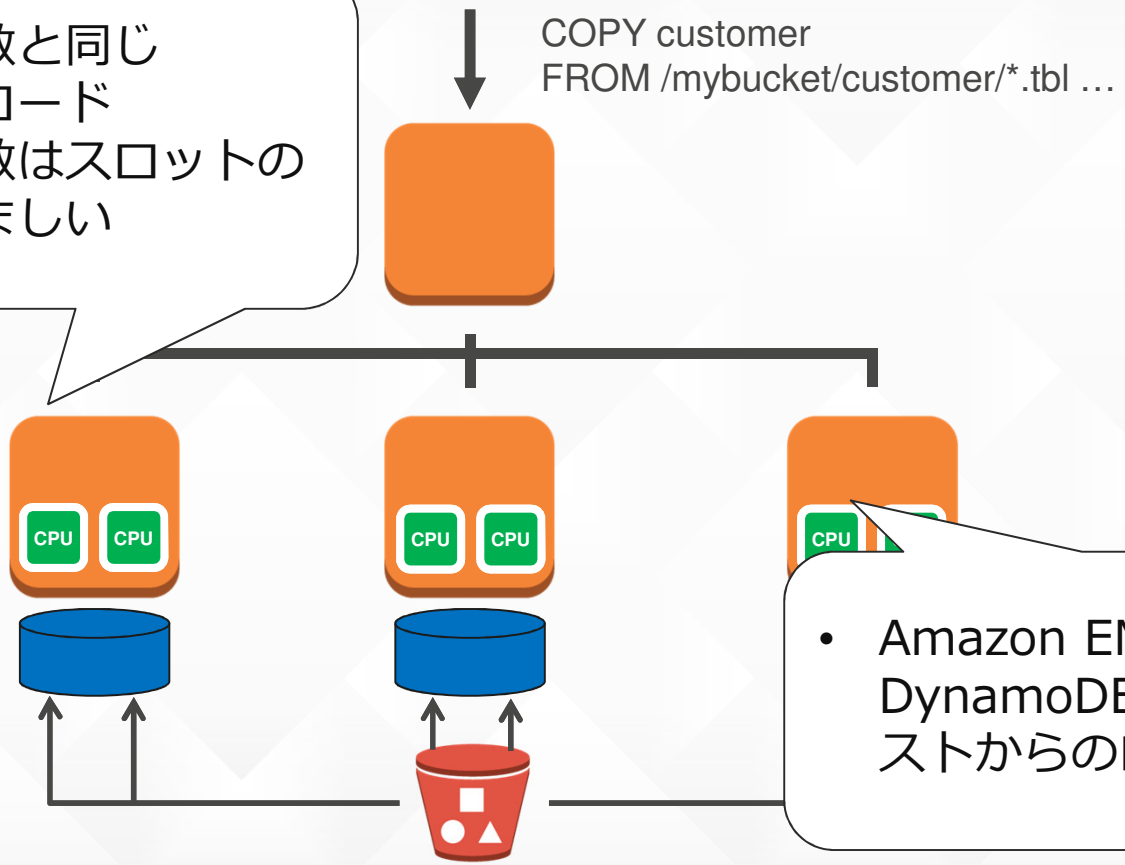
ノードスライス＝
メモリとディスクを
CPUコアと同数に分割
した論理的な単位



amazon web services

アーキテクチャ：データのロード

- スロット数と同じ並列度でロード
- ファイル数はスロットの倍数が望ましい



- Amazon EMR、Amazon DynamoDB、リモートホストからのロードも可能

amazon web services

Redshift
主要アップデート

amazon web services

Redshiftのカスタムドライバ

- Redshiftに最適化されたドライバをリリース
  - JDBC 4.1/4.0をサポート
  - ODBC3.8をサポート、2.xとも互換性あり
- PostgreSQL.orgで公開されているドライバよりもパフォーマンス・信頼性の観点で優れている
- 関連ドキュメント

  http://docs.aws.amazon.com/ja_jp/redshift/latest/mgmt/connecting-to-cluster.html

amazon web services

Interleaved Sort Key（1）

- Sort Keyを複数のカラムに指定が可能になり、フルスキャンを回避し、Disk I/Oを最小限に抑えられる
- 最大８つまでのSort Key列を指定できる

```
CREATE TABLE ~
…
INTERLEAVED SORTKEY (deptid, locid);
```

amazon web services

Sort Key (

DeptId = 1 -> 1 block
LocId = C -> 4 block

DeptId = 1 -> 2 block
LocId = C -> 2 block

Compound Sort Key

DeptId	LocId
1	A
1	B
1	C
1	D
2	A
2	B
2	C
2	D

DeptId	LocId
3	A
3	B
3	C
3	D
4	A
4	B
4	C
4	D

Interleaved Sort Key

DeptId	LocId
1	A
1	B
2	A
2	B
1	C
1	C
2	D
2	D

DeptId	LocId
3	A
3	B
4	A
4	B
3	C
3	D
4	C
4	D

amazon web services

Redshiftにおける
インテグレーションとは？

amazon web services

インテグレーション = データ連携

- オンプレミスとのデータ連携
- AWSサービス間のデータ連携
- 考慮すべき点：
  - シングル vs. パラレル処理？
  - 同期 vs. 非同期実行？
  - どこでデータ変換をすべきか？

amazon web services

オンプレミスとのデータ連携

- データをS3にCSVファイルとして蓄積・保全
- オンプレミス環境からRedshiftへの直接insertは推奨されない
- 大量の同時クライアントからの書込みが発生する場合、DynamoDBやKinesisに書き込みを行い、一定間隔でRedshiftにロード

amazon web services

AWSサービス間のデータ連携

- S3がハブとなり、他のAWSサービスと連携

  例：
  - MySQL内のレコードをS3に出力
  - CSVファイルをRedshiftにロード
  - Webのアクセス・ログをEMRを使って変換
- AWSネイティブのサービスとしては：
  - バッチ処理：Data Pipeline
  - ストリーム処理：Kinesis
  - イベント処理：Lambda

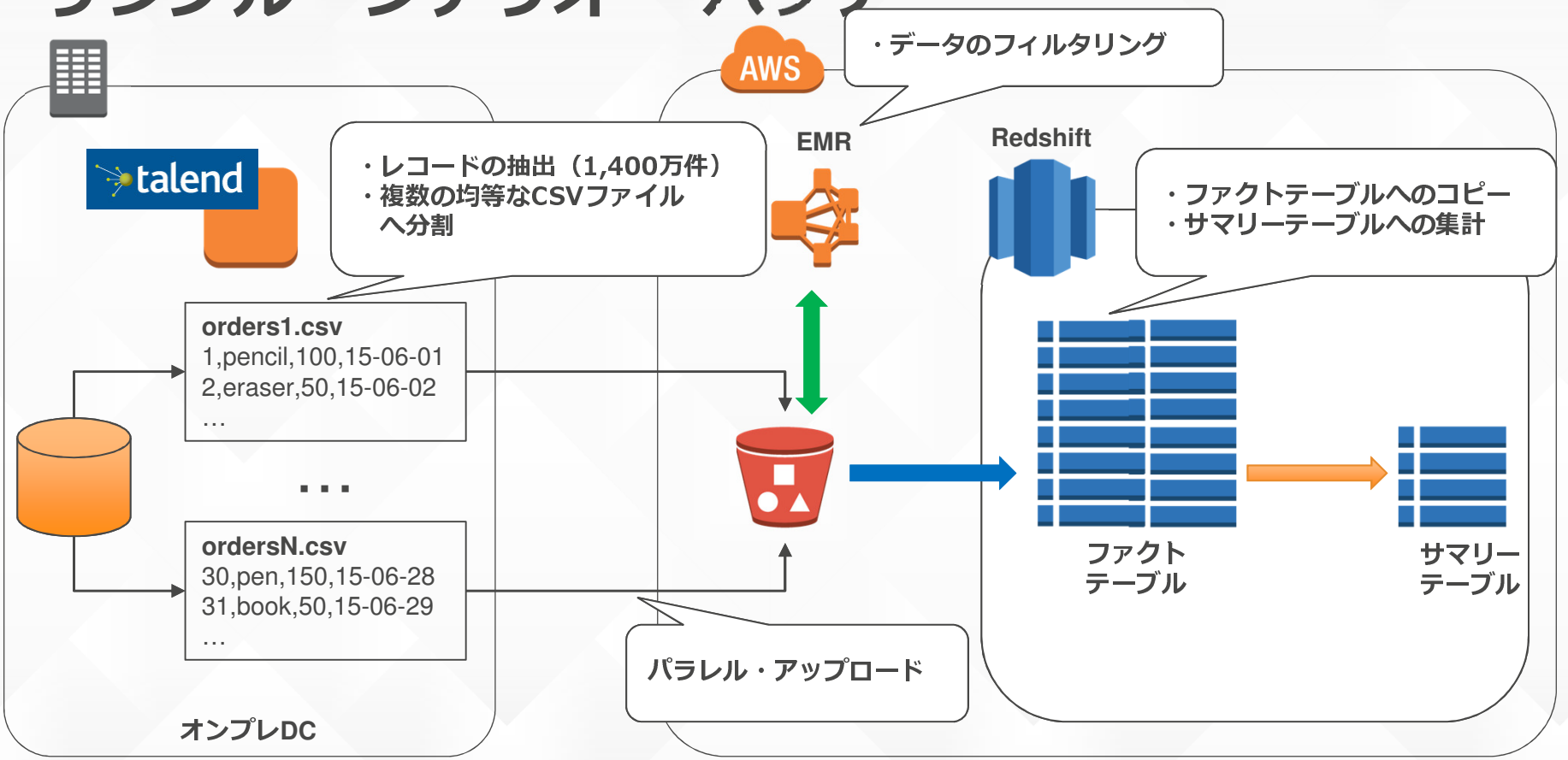
amazon web services

サンプル・シナリオ – バッチ

- RDBMSからデータを抽出（**E**xtract）
- Amazon S3にアップロード（**U**pload）
- Amazon EMRによるデータ変換（**T**ransform）
- Amazon Redshiftにロード（**L**oad）

amazon web services

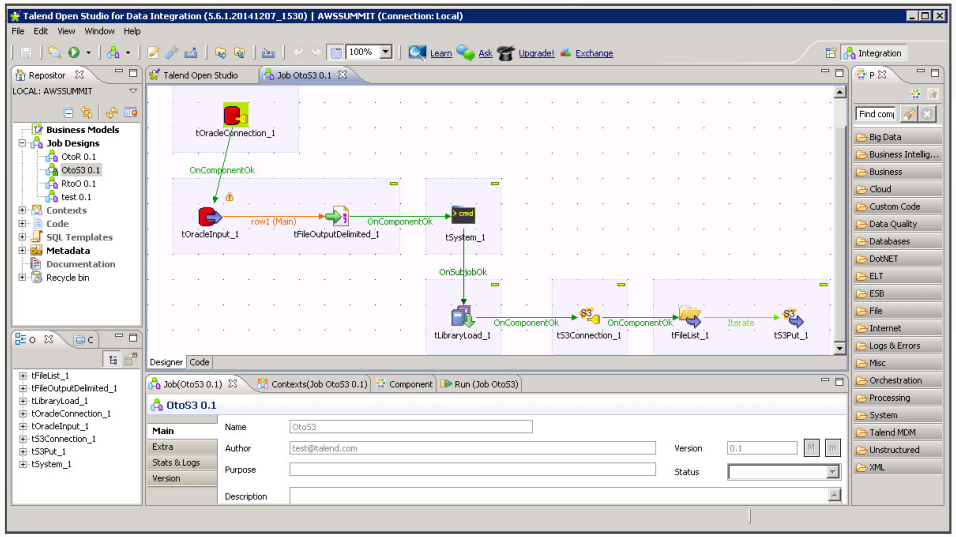
サンプル・シナリオ – バッチ



amazon web services

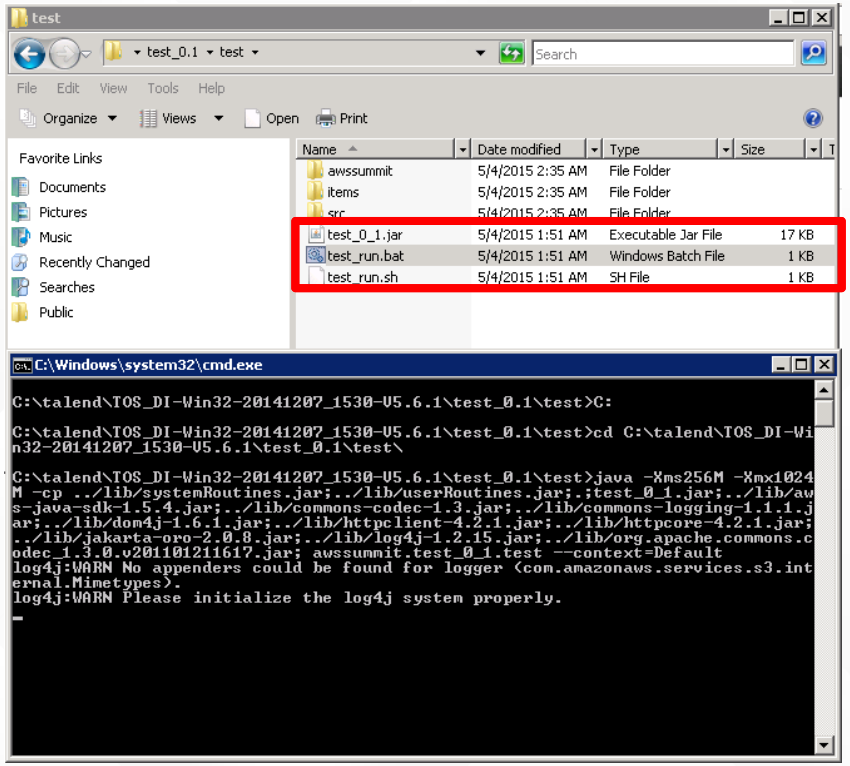
Talendのご紹介（1）

- EclipseベースのETLツール
  - Talend Open Studio for Data Integration
  - OSS、無料でダウンロード可能
- AWS（S3、Redshiftなど）を含む様々なアダプタを標準で実装
- Javaによるカスタム拡張が可能

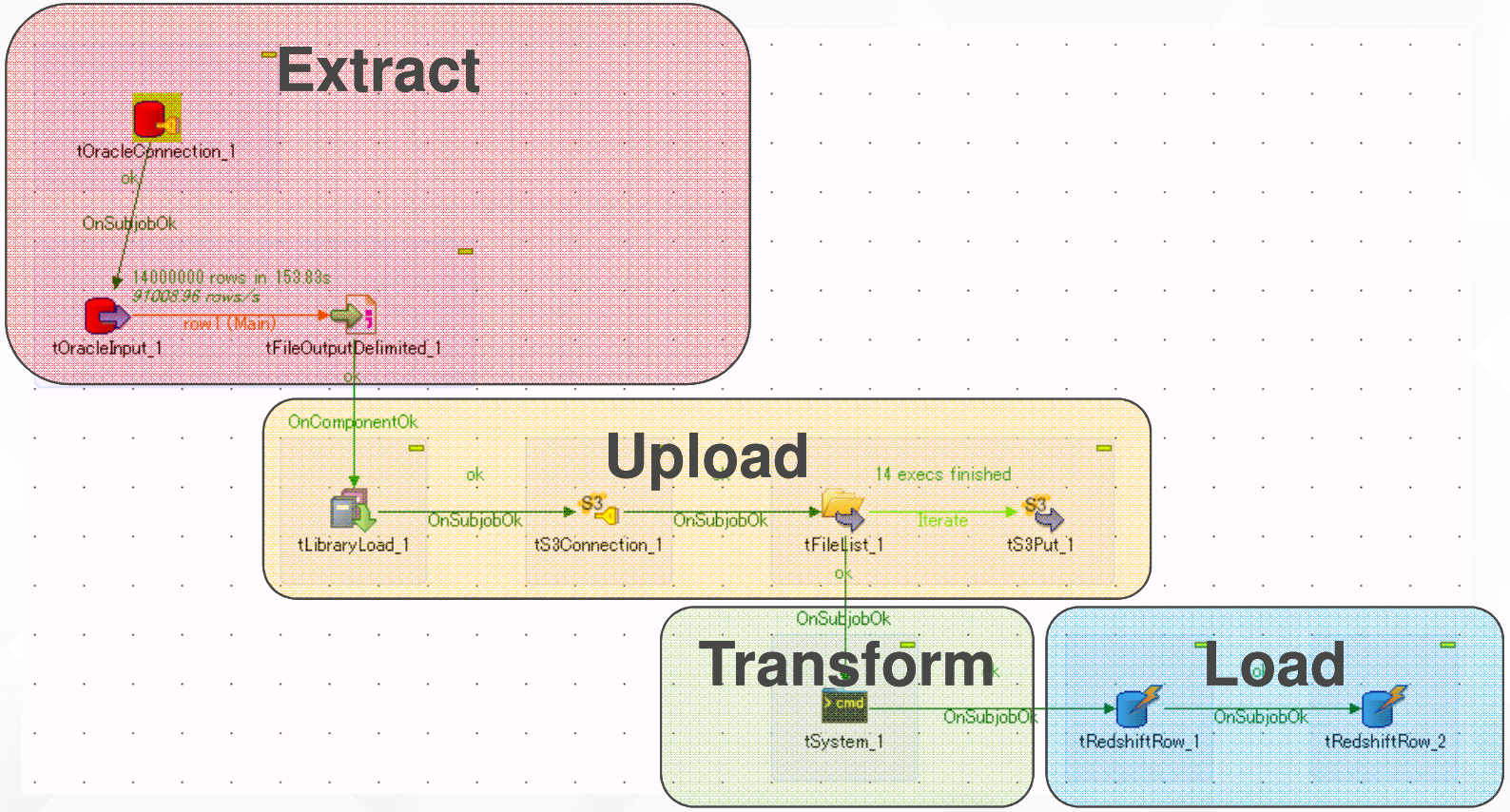


Talendのご紹介（2）

- ジョブを単体のJarにビルド・実行も可能
- スケジューリング、監視、サポート等はEnterpriseバージョンで提供



サンプル・シナリオ – ジョブの定義



Extract（抽出）

amazon web services

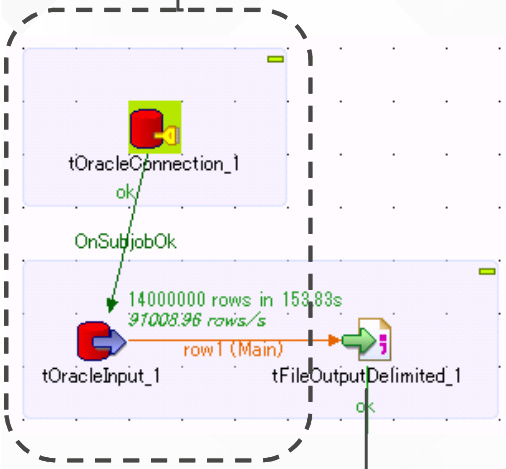
RDBMSからデータの抽出（１）

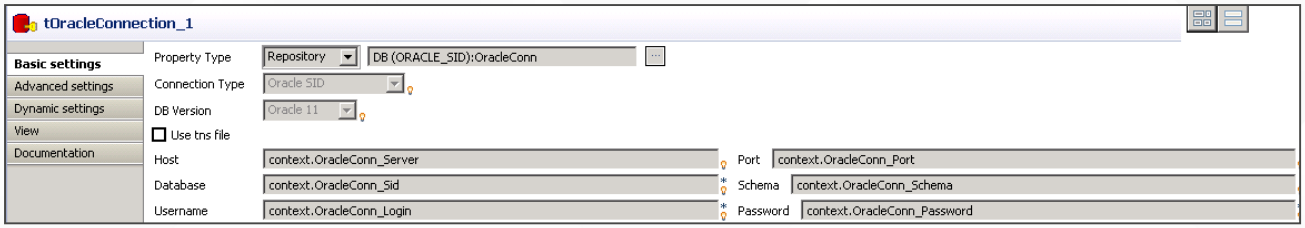
- CSV形式のファイルとして、レコードを抽出
- この時点で複数のファイルに分割
  - ETLツールで抽出条件を含む、SELECT文を記述
  - 一定のレコード数単位で分割
- Redshiftへの並列ロードが可能となる

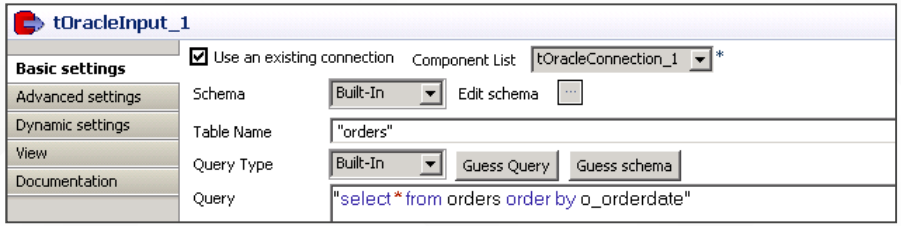
amazon web services

RDBMSからデータの抽出（２）

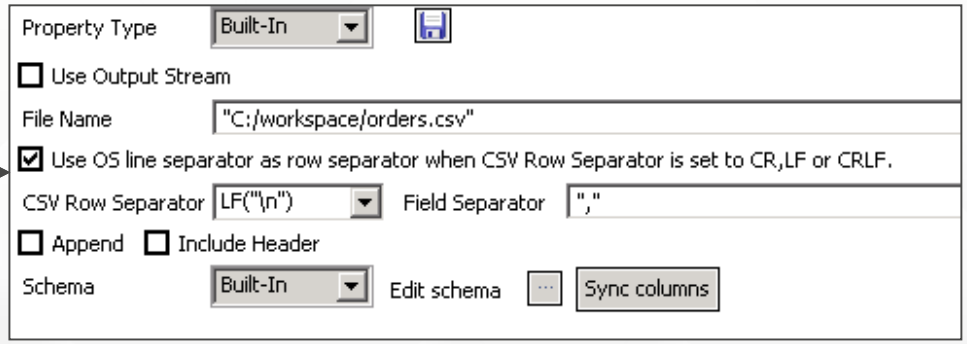
１．Oracleへの接続確立とレコードの抽出







２．CSVファイルの作成



amazon web services

Change Data Capture – CDC（1）

- データのロード先がファクトテーブルの場合、通常は日付や識別子などを抽出条件に使用
  - 当日の売上データだけの抽出
  - 特定の注文番号以降のレコードを抽出
- 差分データの抽出が最も難しい
  - 更新や削除されたレコードの特定
  - 単体のSELECT文だけでは抽出ができない

amazon web services

Change Data Capture – CDC（2）

1. マスター・テーブルのレコード数が少ない場合、差分データを割り出さずに、毎回全件を抽出
2. ソース・テーブルにトリガーを設定し、差分データを変更テーブルに蓄積・抽出
3. RDBMSのトランザクション・ログを解析し、更新ログを抽出できる商用ツールもある

* Talendでは、CDCにEnterpriseバージョンが必要

amazon web services

設計のポイント - Extract

サマリー

- 大量のレコードを抽出する場合、メモリの枯渇を防ぐために、カーソルを利用
- 抽出・分割後、ファイルを圧縮することにより、S3へのアップロード時のコストを削減
- ファイル分割数は、Redshiftクラスタのスロットの倍数が理想

amazon web services

Upload（アップロード）

amazon web services

Amazon S3へのアップロード（1）

- データ量やファイル数に応じて、並列にアップロード
- シリアルなアップロードより、トータルの転送時間を短縮

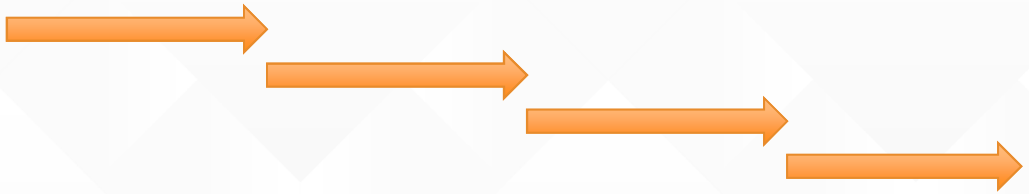
amazon web services

Amazon S3へのアップロード（2）

- 並列アップロードの例（EC2 -> S3）

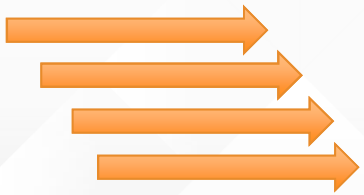
シングル・スレッドでの実行

orders0.csv (135mb)
orders1.csv (135mb)
・・・
orders13.csv (135mb)



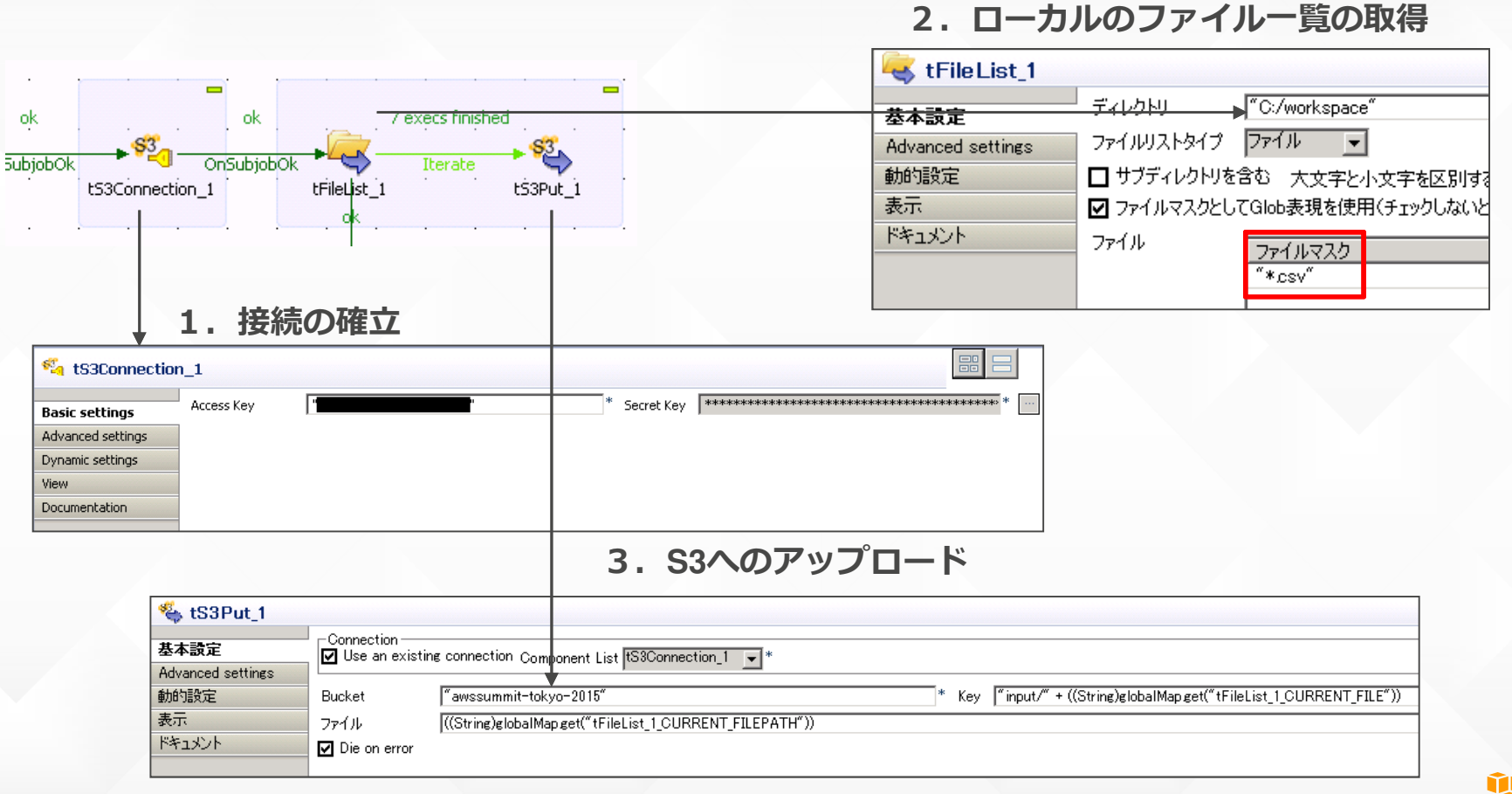
マルチ・スレッドでの実行

orders0.csv (135mb)
orders1.csv (135mb)
・・・
orders13.csv (135mb)



amazon web services

Amazon S3へのアップロード（3）



設計のポイント - Upload

サマリー

- アップロードの並列度を決定する際には、クライアントのスペック（CPUコア数）を考慮
- S3の「キー」先頭４文字をランダム化させるのが性能的には理想だが、通常は管理しやすい「日付」や「テーブル名」毎のフォルダを使用

amazon web services

Transform（変換）

amazon web services

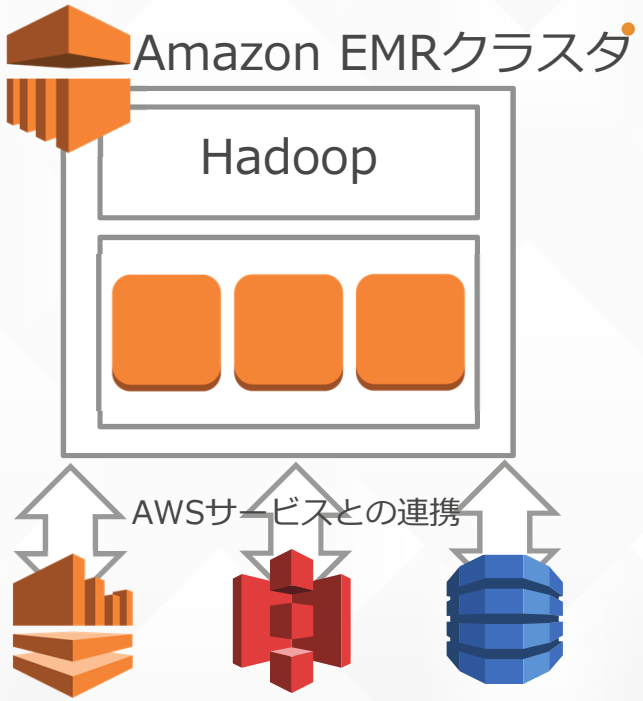
Transformをどこで実行するか？

- AWSにアップロード前のオンプレミス環境
  - ✓ アップロード前にクレンジングし、転送時間の削減
  - ⚠ オンプレ側にリソースが必要
- **S3内のファイルをEMRでバッチ変換**
  - ✓ RedshiftからTransform処理をオフロード
  - ⚠ Hadoop関連の技術を習得
- SQLで一時テーブルから本番テーブルへ
  - ✓ Redshift内でのSQLのみで完結できる
  - ⚠ Redshiftの負荷が増加

amazon web services

Amazon Elastic Map Reduce（EMR）

フルマネージドなHadoopを提供。
運用を気にせずHadoopアプリケーションの開発や利用ができる。



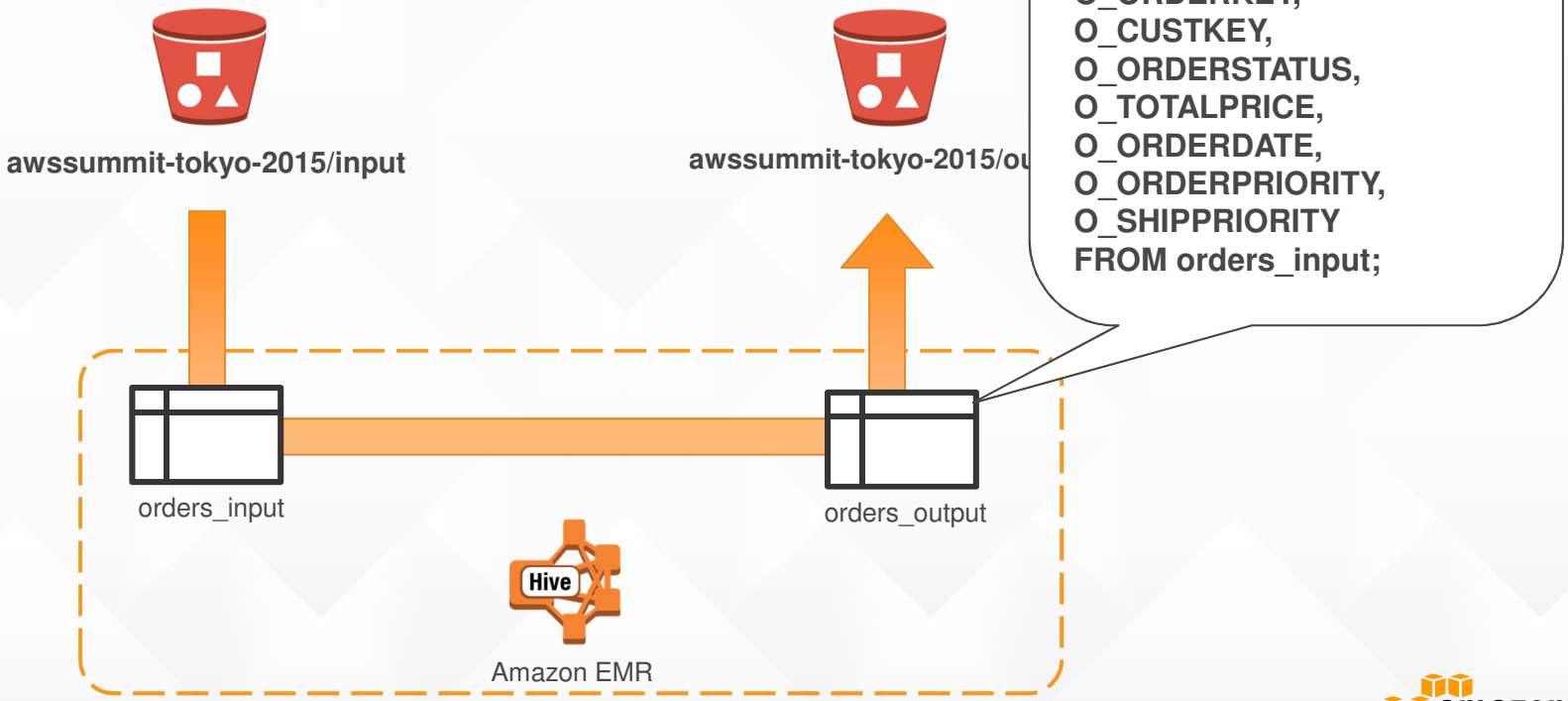
特徴 (http://aws.amazon.com/jp/elasticmapreduce/)

- フルマネージド：クラスタの構築から構成変更、破棄まですべてマネージ
- 自動化：Amazon EMRのAPIを利用するとジョブに合わせてクラスタを起動し、実行させ、終了したらクラスタを破棄、というような自動化が容易
- AWS：Amazon S3やAmazon DynamoDBからデータの入出力が可能

amazon web services

EMRによるデータ変換（１）

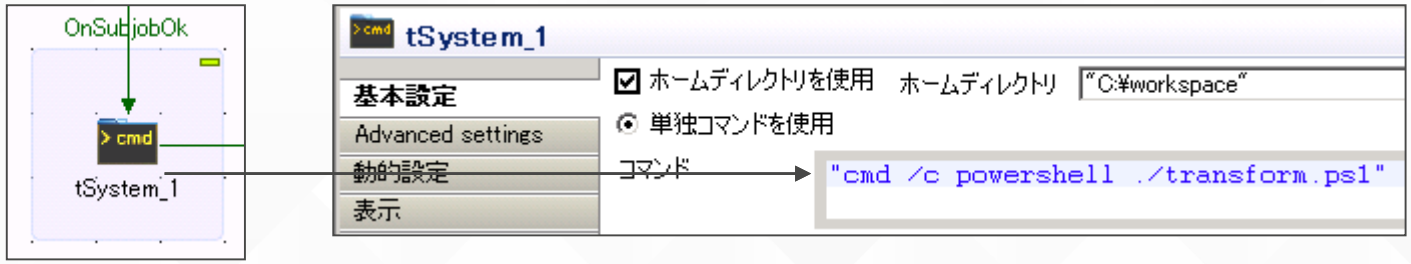
- Hiveによるデータ変換の例



amazon web services

EMRによるデータ変換（2）

- PowershellでAWS CLIとHiveスクリプトを実行



- AWS CLIの呼出しは非同期実行であるため、フローの中で同時実行をさせるには工夫が必要

amazon web services

EMRによるデータ変換（３）

```
$wait = "aws emr wait cluster-running --cluster-id " + $clusterid
$wait_output = invoke-expression $wait

:OUTER for(;;) {
    $describe = "aws emr describe-cluster --cluster-id " + $clusterid
    $describe_output = invoke-expression $describe

    for ($i=0; $i -lt $describe_output.length; $i++)
    {
        if (($i -eq 4) -and ($describe_output[$i] -like "*TERMINATED")){
            break OUTER
        }
    }
    Write-Host "Sleeping..."
    Start-Sleep 40
}
```

クラスタ作成待ち

Hiveスクリプト実行後、クラスタ削除待ち

amazon web services

設計のポイント - Transform

サマリー

- データの変換処理をどこで実行すべきか
- AWSではEMRで下記を容易に実現
  - S3とのデータのIn/Out
  - 変換処理の並列実行
  - クラスタの作成・破棄
- AWS CLIからの呼出しを同期実行するためには、ポーリング等の工夫が必要

amazon web services

Load（ロード）

amazon web services

Redshiftへのロード（1）

- ファイル一覧や正規表現をCOPYコマンドに指定すると、コンピュート・ノードが並列にデータをロード
- 単一ファイルの指定では、特定のコンピュートノードのみがS3にアクセスするため、並列ロードにならない
- 新規テーブルの場合、明示的に圧縮アルゴリズムを指定し、サンプリング処理を回避

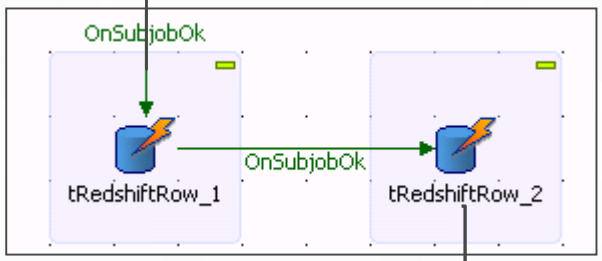
amazon web services

Redshiftへのロード（2）

- ロードに失敗したレコードは、「stl_load_errors」表に格納される
- 「MAXERRORS」オプションにより、許容できる最大エラー数（レコード数）を指定

amazon web services

Redshiftへのロード（3）



1．COPYコマンドの実行

tRedshiftRow_1

Basic settings / Advanced settings / Dynamic settings / View / Documentation

Property Type: Built-In
Use an existing connection
Host: "mycluster.cgpeksjvvvjq.ap-northeast-1.redshift.amazonaws.com"
Port: "5439"
Database: "mydb"
Schema: "public"
Username: "awsuser"
Password: **********
Schema: Built-In　Edit schema　Table Name: ""
Query Type: Built-In　Guess Query
Query:

```
"copy orders from 's3://awssummit-tokyo-2015/orders' CREDENTIALS
'aws_access_key_id=[redacted]aws_secret_access_key=[redacted]'
CSV QUOTE '\"'
DATEFORMAT 'DD-MM-YYYY'
```

2．INSERT ~ SELECTの実行

tRedshiftRow_2

基本設定 / Advanced settings / 動的設定 / 表示 / ドキュメント

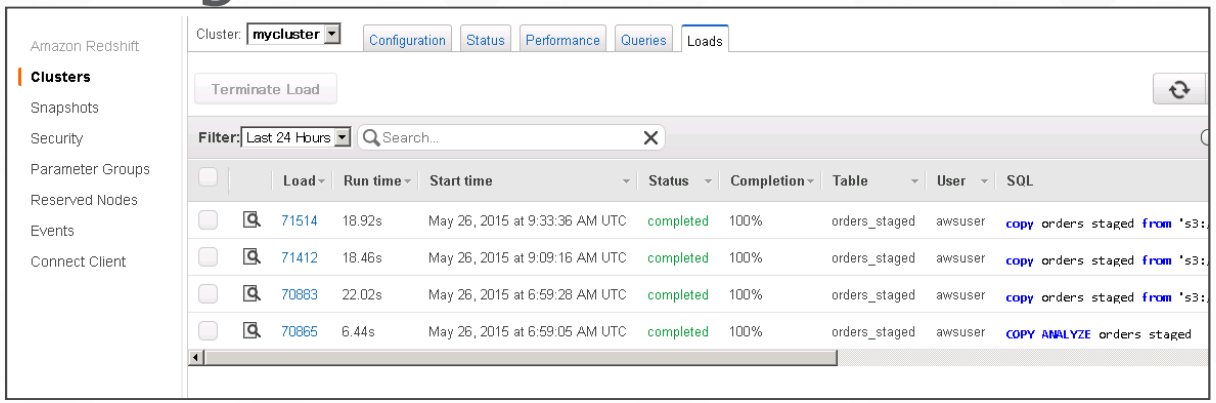
プロパティ タイプ: 組み込み
既存の接続を使用
ホスト: "mycluster.cgpeksjvvvjq.ap-northeast-1.redshift.amazonaws.com"
ポート: "5439"
データベース: "mydb"
スキーマ: "public"
ユーザ名: "awsuser"
パスワード: **********
スキーマ: 組み込み　スキーマの編集　テーブル名: ""
クエリタイプ: 組み込み　Guess Query
クエリ:

```
"insert into orders_summary select o_shippriority, sum(o_totalprice) from orders group by o_shippriority"
```

エラーで停止

ロードの確認

- Management Consoleからの参照



- STLテーブルの参照

  select query, trim(filename) as file, curtime as updated
  from **stl_load_commits**;

amazon web services

設計のポイント - Load

サマリー

- ロード時には、テーブル・ロックがかかる
  - 対象テーブルへのアクセス頻度が低いタイミング
- ロード先のテーブルを本番テーブルと差し替え
  - alter table orders_temp rename to orders_prod;
- 「INSERT ～ SELECT」文は、COPY同様にコンピュート・ノードで並列処理される

amazon web services

まとめ

amazon web services

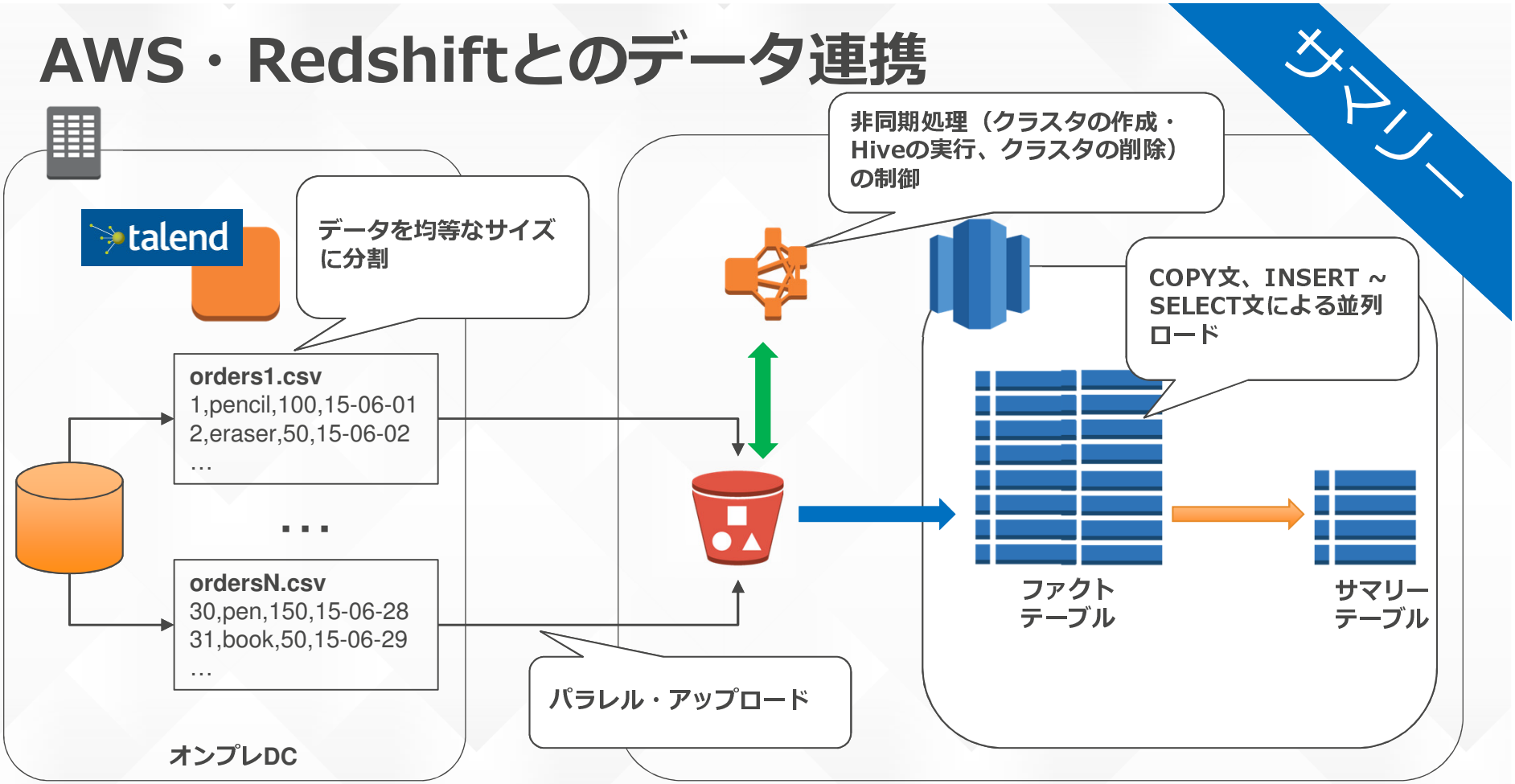
Talendについて

サマリー

- ✓ ツール導入によるETL実装の効率化
  - DB接続情報を共通プロパティ化 -> ジョブ間で共有
  - 実装段階では、毎回全フローを実行するのではなく
    各ステップを局所的に実行・デバッグ
- ✓ スタンドアローンで動作するJavaプログラムとしてビルド・実行
- ⚠ ツール自体の学習が必要
- ⚠ RedshiftドライバやEMRなどの対応

amazon web services

AWS・Redshiftとのデータ連携



amazon web services

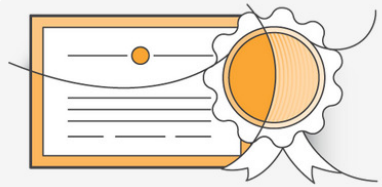
AWSトレーニング @ AWS Summit Tokyo

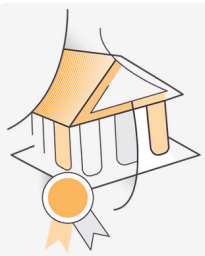
セルフペースラボ：@パミール1F 瑞光

AWS クラウドに実際に触れてみませんか？
ご自分の AWS アカウントをおつくりいただけなくても、
AWS クラウドを体験いただけます。

AWS認定試験（有償）：@ パミール1F 黄玉

特設認定試験会場を AWS Summit Tokyo 2015 会場に開設
Devopsエンジニア-プロフェッショナル認定試験を先行受験いただけます。





AWS認定資格者取得専用ラウンジ：@ パミール1F 青玉

他の AWS 認定資格をお持ちの方とのネットワーキングにぜひラウンジをご活用
ください。
お席や充電器、お飲物などを用意し、皆様をお待ちしております。

amazon web services

ご清聴ありがとうございました。

amazon web services