AWS Black Belt Online Seminar

AWSにおける Hudi/Iceberg/Delta Lake の使いどころと違いについて

Chie Hayashida

Solution Architect
2023/05/16

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、 AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
  - https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/
  - https://www.youtube.com/playlist?list=PLzWGOASvSx6FIwIC2X1nObr1KcMCBBlqY

ご感想は Twitter へ！ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では 2023年 5 月時点のサービス内容および価格についてご説明しています。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (https://aws.amazon.com/) にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS 公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっています。日本居住者のお客様には別途消費税をご請求させていただきます
- 技術的な内容に関しましては、有料の AWS サポート窓口へお問い合わせください
- 料金面でのお問い合わせに関しましては、カスタマーサポート窓口へお問い合わせください (マネジメントコンソールへのログインが必要です)

自己紹介

林田 千瑛
Chie Hayashida

アマゾン ウェブ サービス ジャパン合同会社
ソリューションアーキテクト

データ分析技術からデータ活用まで
幅広くご支援します！

従来のデータレイクでよく利用されるフォーマット

- CSV, JSON
  - 人間が手動で操作しやすい
  - ビジネスアプリケーションなどと連携しやすい
- AVRO（行指向フォーマット）
  - データベースが高速に処理しやすい
  - 行ごとにエンコードされているため、行単位での操作に優れているため、OLAP処理に利用可能
  - Schema Evolutionなど多様な機能がある
- Parquet、ORC（列指向フォーマット）
  - 列ごとにエンコードされているため、列単位でスキャンするような分析用途に適している
  - 列ごとに圧縮することで対象がパターン化するため効率よく圧縮できる

データレイクで推奨される
フォーマット

従来のデータレイクの例とその特徴

大量データに対するテーブルやパーティションレベルでの
書き込み・読み込みが得意な一方、OLTP/OLAP的なレコードレベルの
書き込み・読み込みが不得意

従来のデータレイクの技術課題

- スモールファイルによるパフォーマンス劣化
- データの編集・削除が苦手
- 同時アクセス時の整合性を担保できない
- データが追加・更新されると過去の状態が復元できない

共通的なユースケース

- ストリームパイプライン
- CDC（Change Data Capture）によるデータやスキーマの変更
- 個人情報保護
- データサイエンス

ユースケース1：ストリームパイプライン

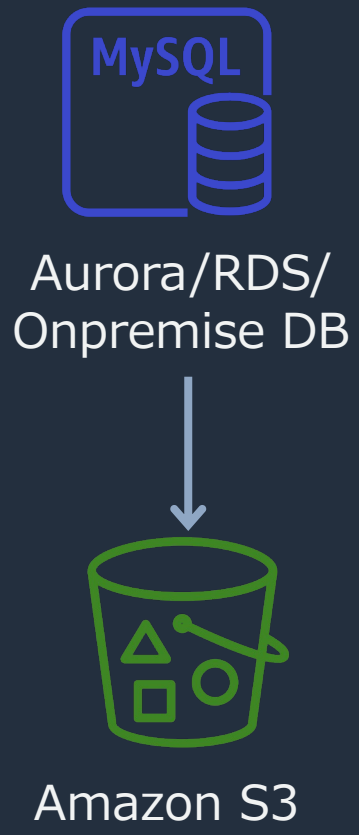
- 書き込みと読み込みの両方のパフォーマンス担保
  - データの保存間隔を短くするとスモールファイルによる読み込み性能劣化を避けたい
- 重複データにより分析が複雑化を避けたい
- 遅延イベントのハンドリング
  - 到着時刻によるオーダリングが必要な場合がある
- ソースデータのスキーマ変更への追従
- 上記に対応するためにバッチとストリームの両方のパイプライン保持することによるコスト増（ラムダアーキテクチャ）

Kinesis Firehose/
Glue Streaming

Amazon S3

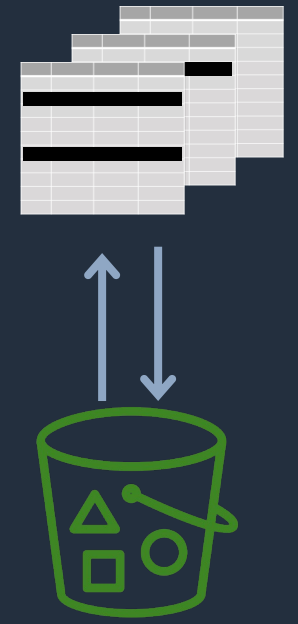
ユースケース2：CDC（Change Data Capture）

- S3上のターゲットテーブルへUPSERTを行う必要がある
- 分析時のトランザクション担保



ユースケース3：個人情報保護

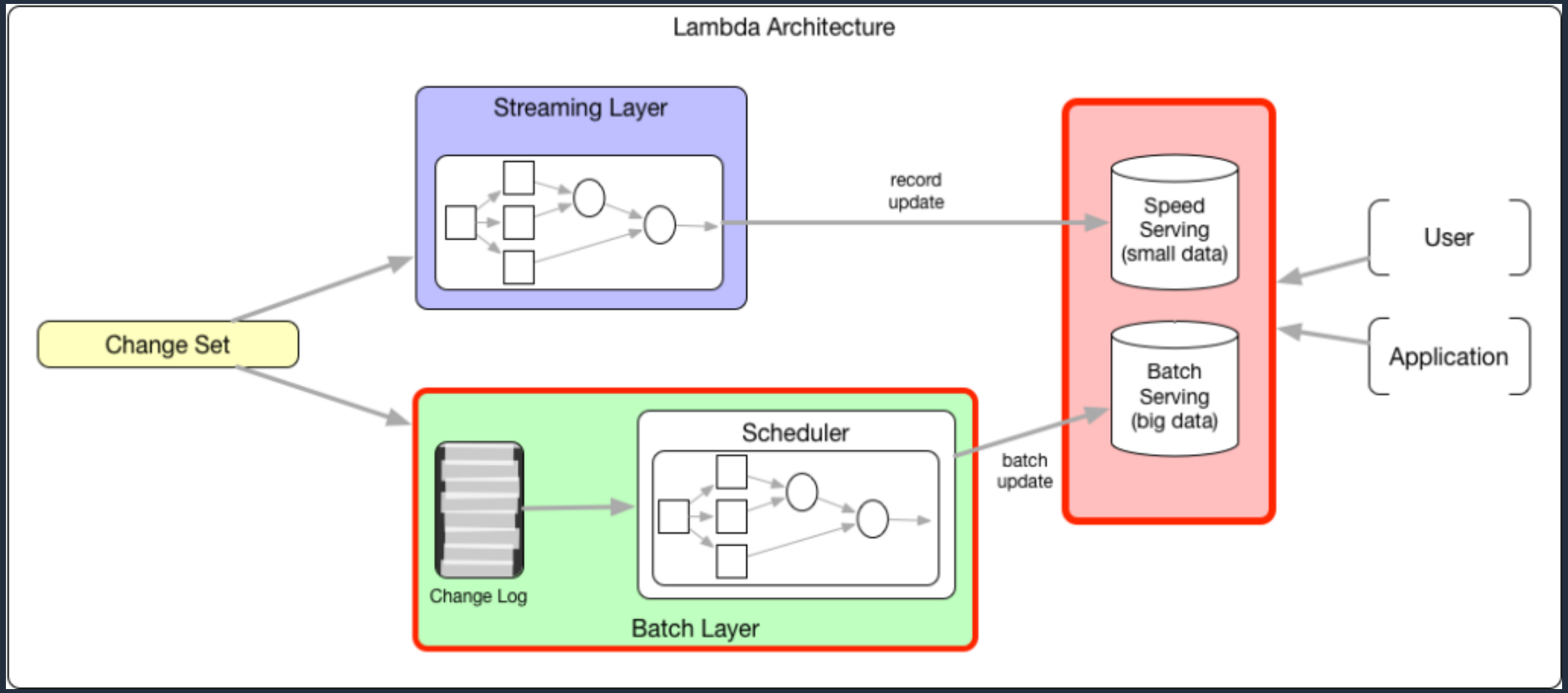
- 決まった時間内に特定の個人情報（主にレコードベース）を削除する必要がある
- データ削除中のトランザクションが担保



ユースケース4：データサイエンス

- データサイエンスでは、実験を再現できることが重要
  - ある特定の時点のデータを利用した分析を再現できる必要がある
  - 現在と過去のデータを比較したい場合がある
  - エラーが発生した際の調査
- 上記実施のためにエンジニアリングコストをかけたくない
  - スナップショットから復元などはやりたくない

従来のソリューション – ラムダアーキテクチャ



アーキテクチャが複雑化してメンテナンスしにくいデメリットがある

https://www.oreilly.com/radar/questioning-the-lambda-architecture/

【再掲】従来のデータレイクの技術課題

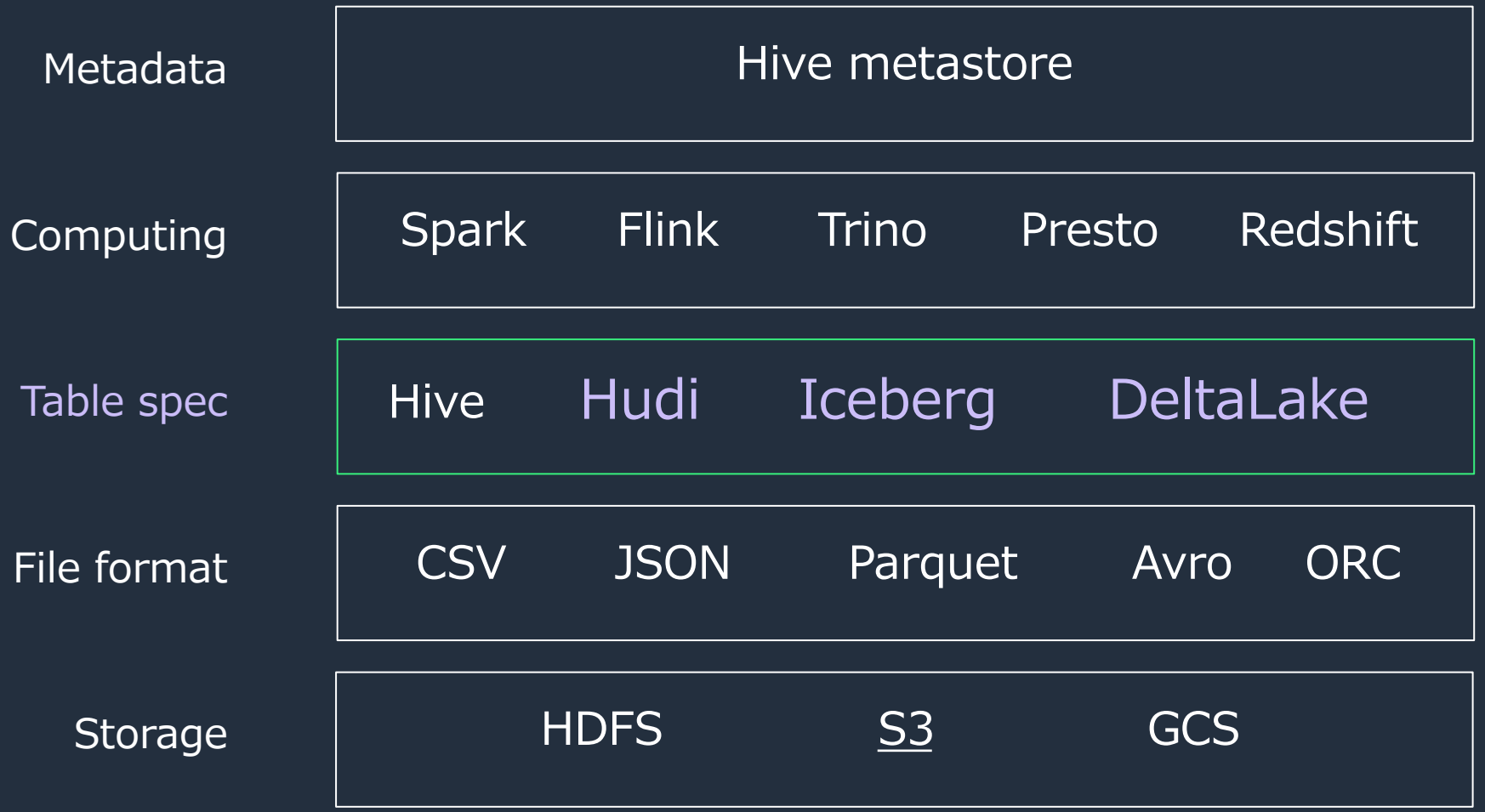
- スモールファイルによるパフォーマンス劣化
- データの編集・削除が苦手
- 同時アクセス時の整合性を担保できない
- データが追加・更新されると過去の状態が復元できない

新たなデータレイクフォーマット＋既存クエリエンジンによるサポートによって解決

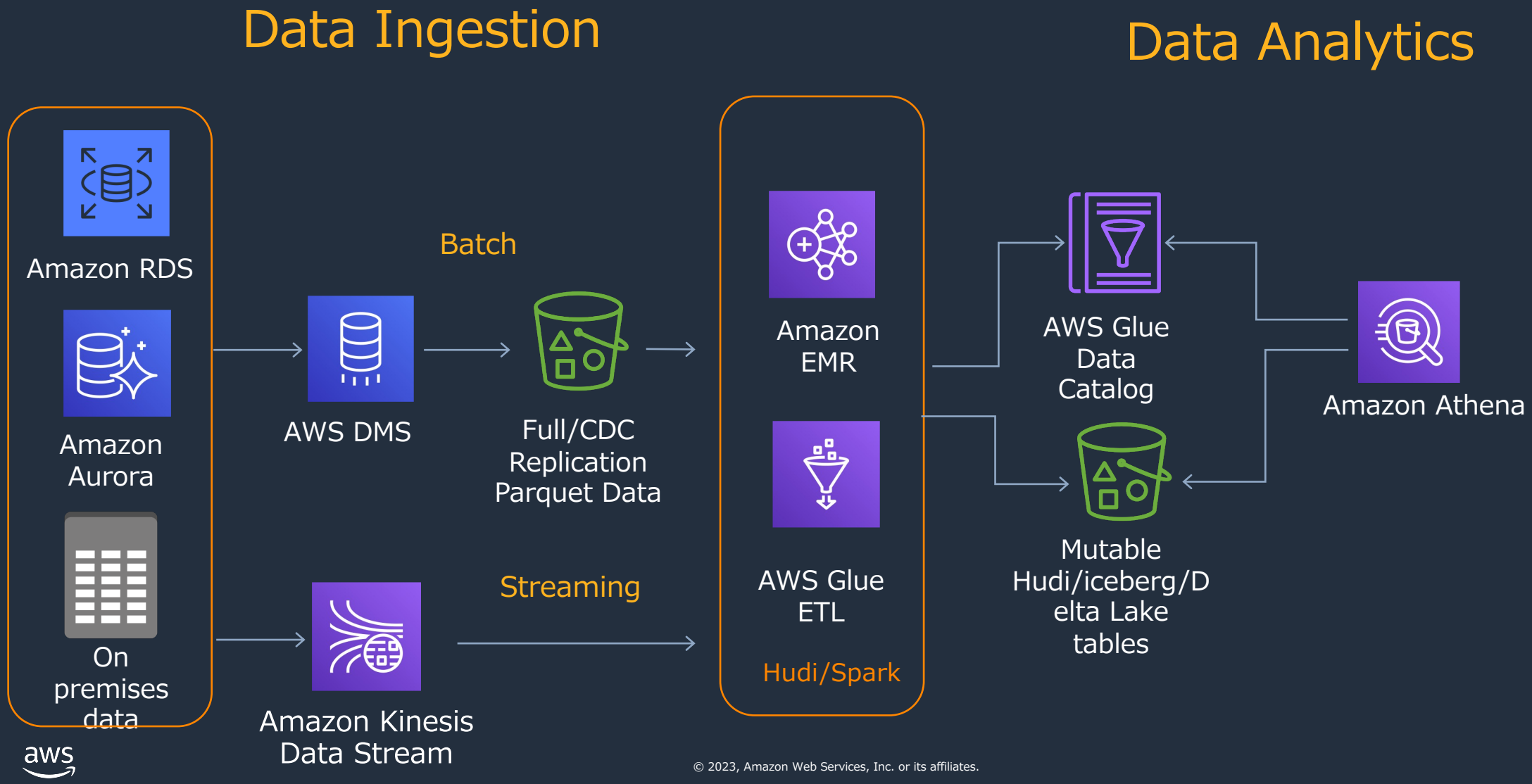
新たなデータレイクフォーマット

- Apache Hudi
- Apache Iceberg
- Delta Lake

データレイクフォーマットの構成



データレイクフォーマットを利用したパイプラインの例



4つのソリューションで共通する考え方

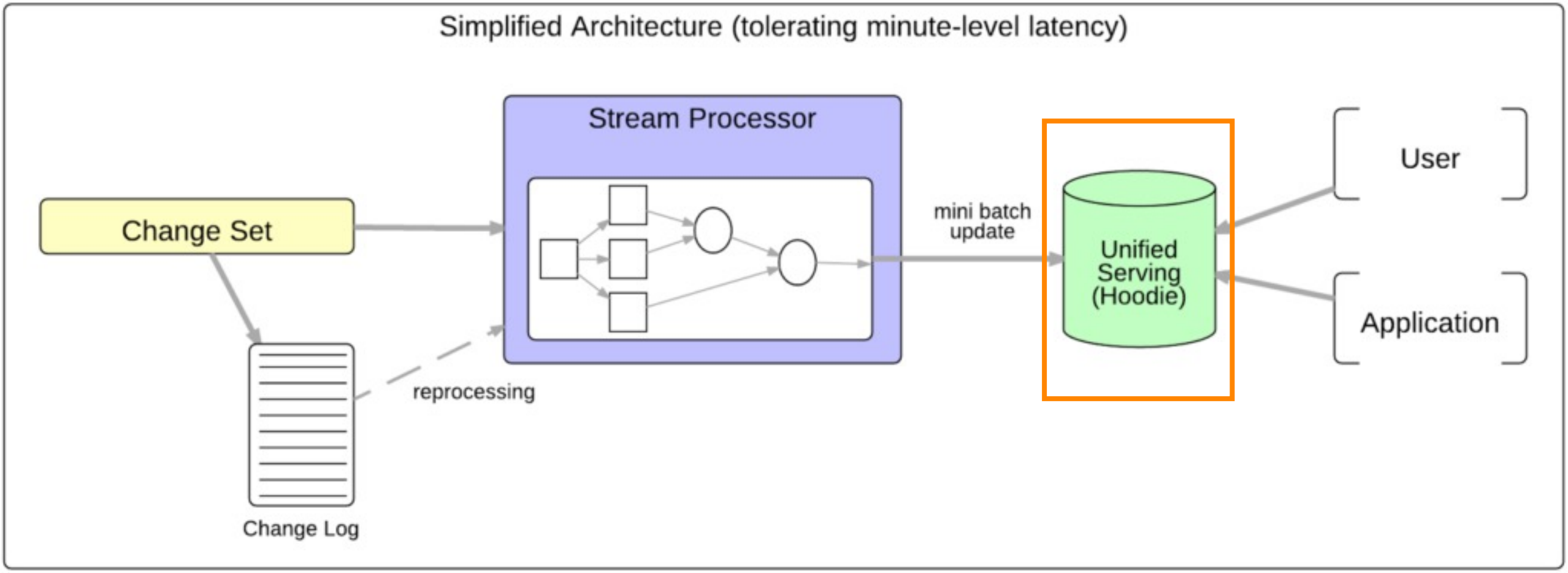
- 新しいデータ・フォーマットと、そのフォーマットの読み書きのためのライブラリ/ユーティリティを提供することで、従来のクエリエンジンによるソリューションを提供
- 従来のHiveメタストア上のメタデータに加えて追加のメタデータを利用してトランザクションやタイムトラベルを実現可能にする
- インデックスや統計情報を利用してパフォーマンスを改善する

Apache Hudi

Apache Hudiとは

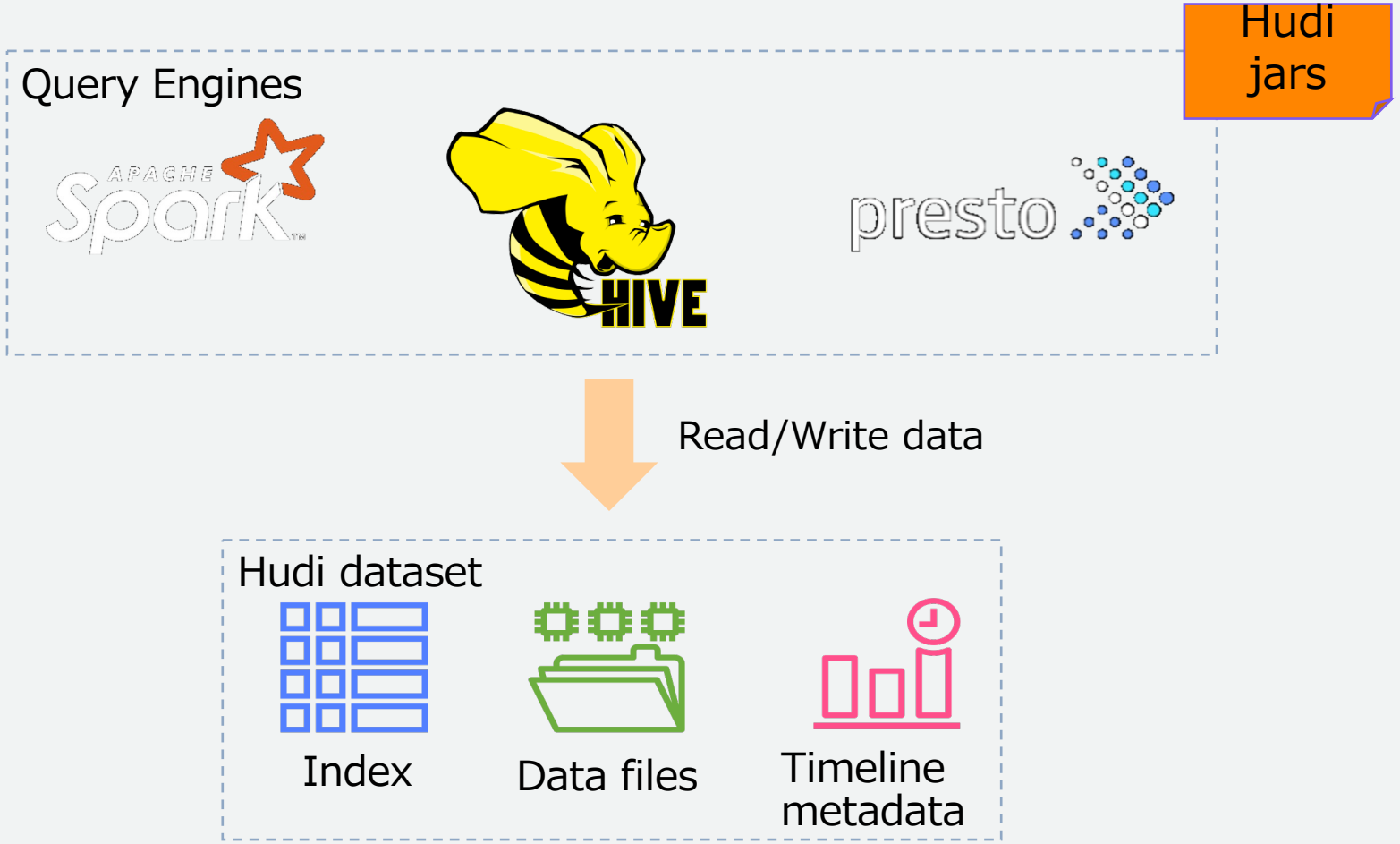
- 2016年からUberによって開発された、最初のオープンソースのフォーマット
- Hadoop Upsert Delte and Incremental の略
- 現在はOneHouseが出資
- Spark, Trino, Flink, Presto, Hive and Impala などでサポート
- データファイルはParquetとAvroで構成される
- 2023/05現在最新バージョンは0.13.0

UberにおけるHudi利用のアーキテクチャ

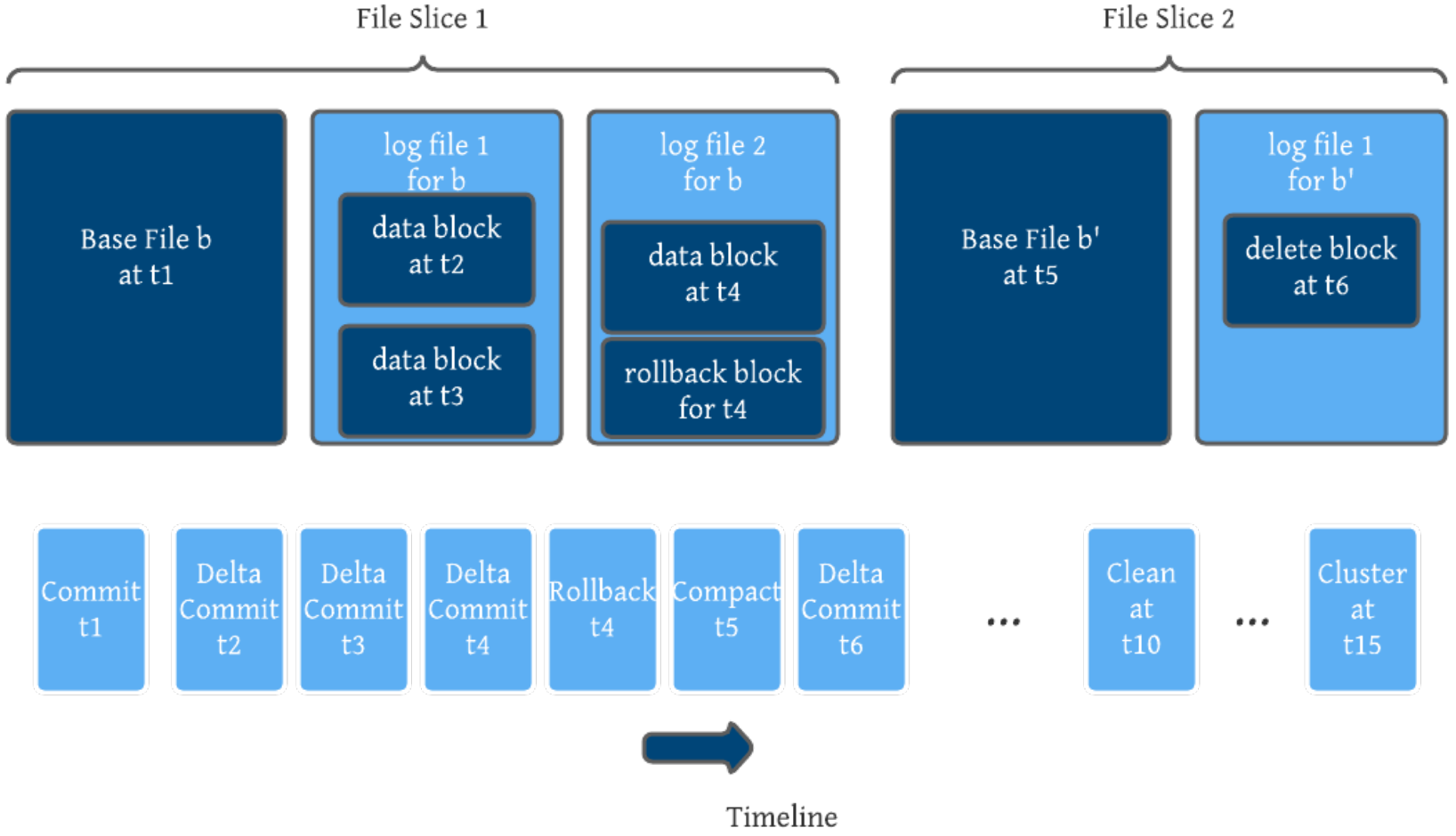


https://eng.uber.com/hoodie/

Apache Hudi の構成



Hudi file format



Source: https://hudi.apache.org/blog/2021/07/21/streaming-data-lake-platform

Apache Hudi の2つのテーブルタイプ

Copy On Write

Read heavy

Columnar only

新規データは既存のParquetファイルに統合

Hudi Dataset

Merge On Read

Write heavy

Columnar + row-based

新規データはAvroで一時保存し、決まったタイミングでParquetに統合（Compaction）

Streaming Ingestion等のユースケースで利用

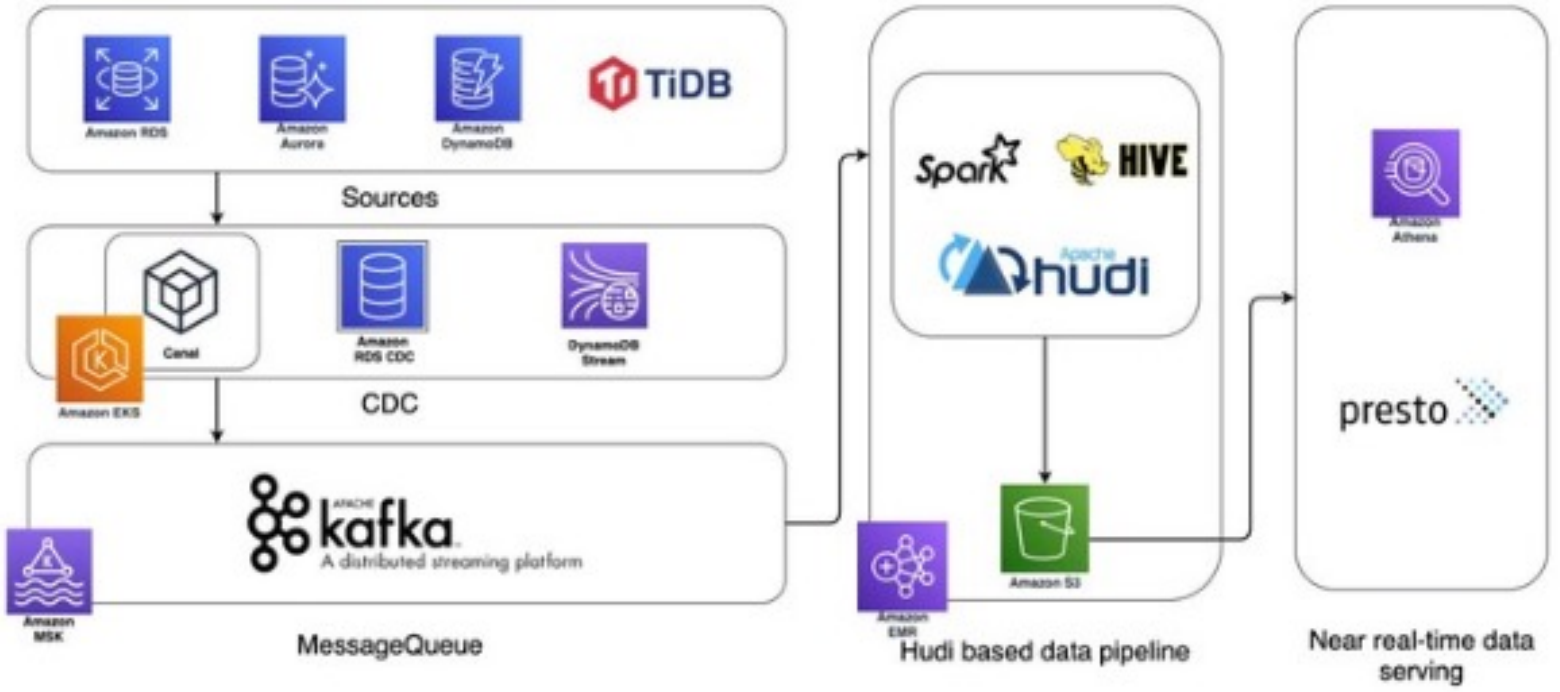
Storage Trade-off

Trade-off	Copy-on-Write	Merge-on-Read
Data latency	Higher	Lower
Update cost (I/O)	Higher (rewrite entire Parquet)	Lower (append to delta log)
Parquet file size	Smaller (high update cost)	Larger
Write amplification	Higher	Lower (depending on compaction strategy)

Data Lake at PayPay

PayPay

Data Change Capture based Pipeline using Apache Hudi: *'New Pipeline'*



- Table size does not matter
- No additional index required
- No missing data
- Data is no more stale
- Capture all DDL/DML events on table

https://speakerdeck.com/paypaycorporation/near-real-time-data-lake-at-paypay?slide=11

用途に応じた3種類のビュー

- Read Optimized View
  - リアルタイム性を多少犠牲にして読み込みのパフォーマンスを重視したビュー
  - 大量データを従来の列指向フォーマットと遜色ないパフォーマンスで読み込める
- Realtime View (MoRのみ)
  - 最新データを閲覧可能なビュー
- Incremental View
  - 差分・増分データを閲覧可能なビュー
  - 差分・増分データの転送などに利用

AWSアナリティクスサービスの対応状況 （2023/05/16 現在）

- **Amazon EMR**
  - 最新のEMR 6.10.0 ではHudi 0.12.2-amzn-0 をサポート
  - 上記以外のバージョンもJARをロードすることで利用可能
- **Amazon Athena**
  - Athena Engine 3では Hudi 0.12.2をサポート（Hudiは後方互換性を担保）
  - 増分クエリ未サポート、CTASやINSERT INTO未サポートなどの制限あり※
- **AWS Glue**
  - Glue 4.0ではHudi 0.12.1 をネイティブサポート
  - 上記以外のバージョンもJARをロードすることで利用可能
  - GlueカタログはHudiテーブルをETLジョブにて登録可能（2023/05現在クローラはHudi未サポート）
- **Amazon Redshift**
  - 0.5.2, 0.6.0, 0.7.0, 0.8.0, 0.9.0, 0.10.0, 0.10.1, 0.11.0, 0.11.1のCoW形式のデータを外部表でサポート
- **Amazon Kinesis Data Analytics**
  - JARをロードすることで任意のバージョンを利用可能

※https://docs.aws.amazon.com/ja_jp/athena/latest/ug/querying-hudi.html

Apache Iceberg

Apache Icebergとは

- 2017年からNetflixによって開発されたオープンソースのフォーマット
- 現在はTabularが出資
- Spark, Trino, Flink, Presto, Hive, Snowflake などでサポート
- データファイルはParquet、Avro、ORCで構成される
- 2022/10 に1.0 がリリース
- 2023/05 現在最新バージョンは1.2.1

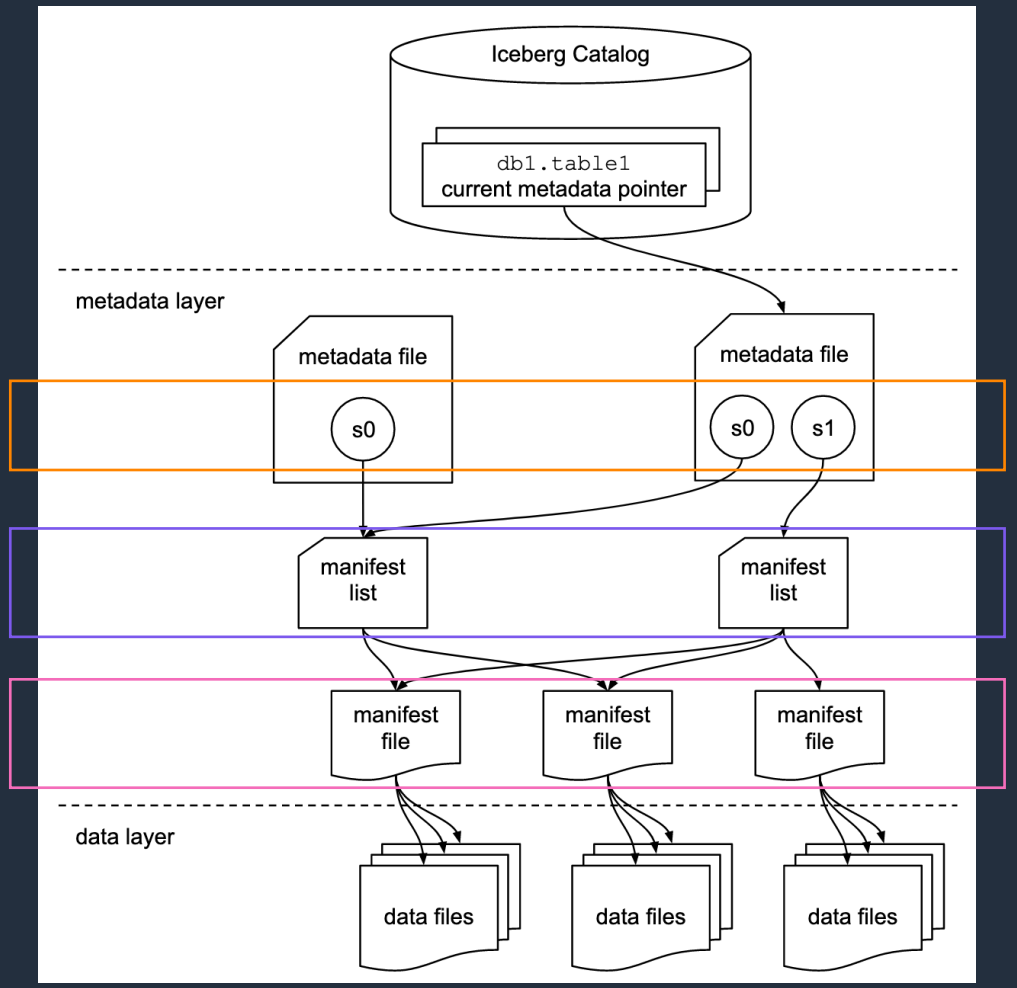
Apache Icebergの構成

A metadata file store metadata about a table at a certain point in time.

A snapshot is the state of a table at some time.

A manifest list is a metadata file that lists the manifests that make up a table snapshot.

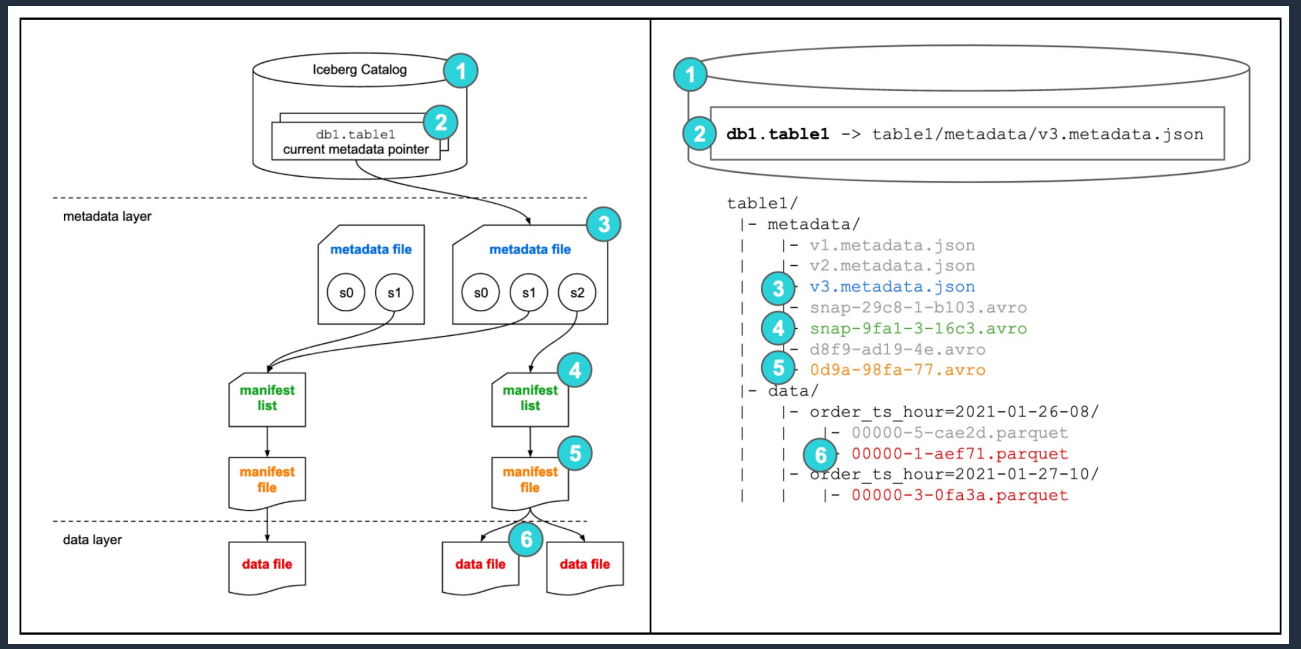
A manifest file is a metadata file that lists a subset of data files that make up a snapshot.



https://iceberg.apache.org/spec/

Faster scan planning, O(1) manifest reads

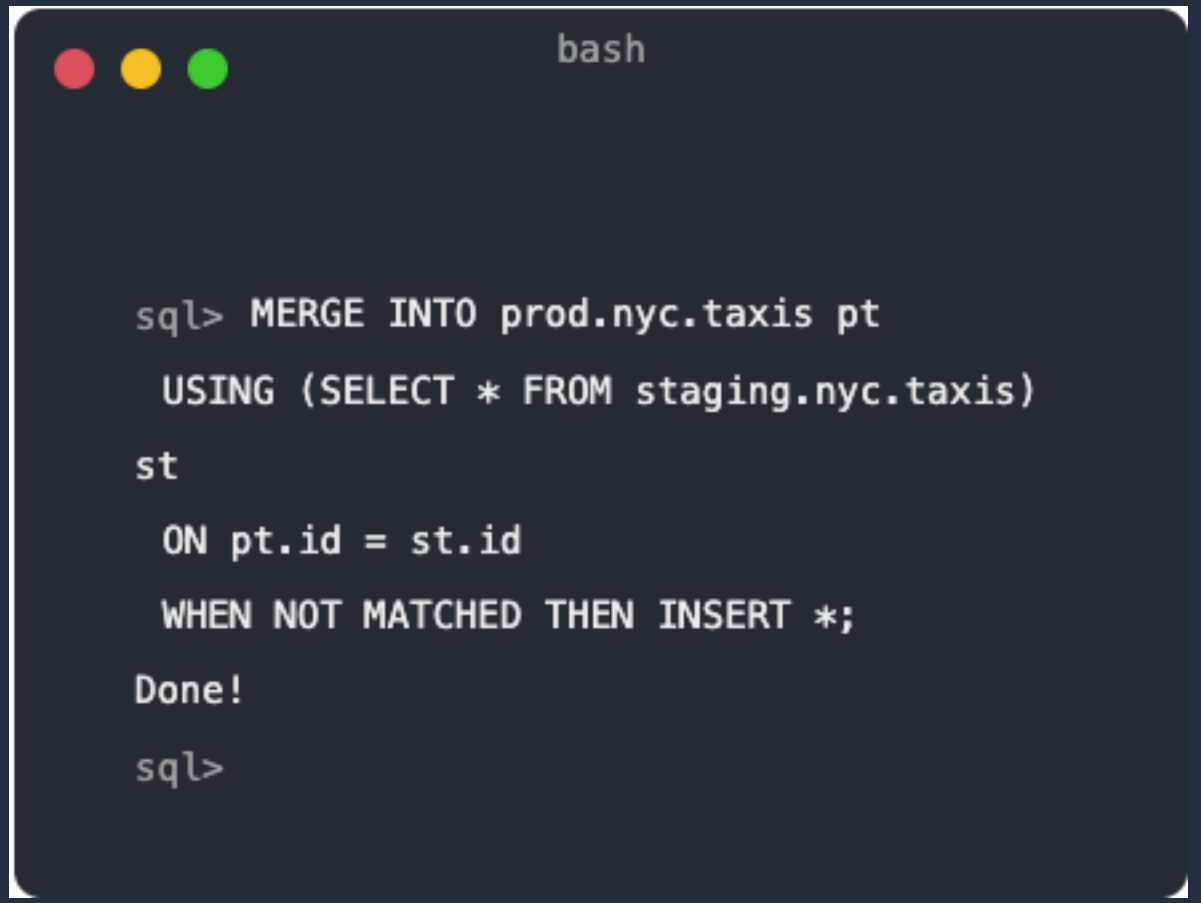
```
SELECT *
FROM table1
WHERE order_ts = DATE '2021-01-26'
```



https://www.dremio.com/resources/guides/apache-iceberg-an-architectural-look-under-the-covers/

Expressive SQL

- 新規データのマージ、既存行の更新、データ削除などを実行するための柔軟なSQLをサポート



Evolution

- Full Schema Evolution
  - スキーマ変更の機能が豊富。メタデータ変更のみなので高速に処理可能
- Partition Evolution
  - Partition Specを更新することでパーティションレイアウトを途中から変更可能
  - Hidden Partitioning
    - パーティションを自動で生成し、レイアウトを改善する。
    - ユーザはパーティション列に合わせたクエリを書かなくとも高速にクエリを処理できる
  - Sort Order Evolution
    - ソート順を途中から変更可能

Apache Icebergの特徴

- タイムトラベルとロールバック
  - スナップショットIDとタイムスタンプのどちらかを指定してクエリ可能
  - バージョンロールバックによりテーブルを迅速に正常な状態に戻すことが可能
- コンパクション
  - Bin-packingやソートなど様々な戦略を選択して
    ファイルレイアウトとサイズを最適化できる
- CoW と MoR を操作によって切り替え可能

bash

```
sql> CALL
system.rewrite_data_files("nyc.taxis");
```

bash

```
scala> spark.read.table("taxis").count()
2,853,020
scala> val ONE_DAY_MS=86400000
scala> val NOW=System.currentTimeMillis()
scala> val YESTERDAY=NOW - ONE_DAY_MS
scala> (spark
 .read
 .option("as-of-timestamp", YESTERDAY)
 .table("taxis")
 .count())
2,798,371
```

AWSアナリティクスサービスの対応状況 （2023/05/16 現在）

- **Amazon EMR**
  - 最新のEMR 6.5.0以降 でIcebergを公式サポート。最新のEMR6.10.0ではIceberg 1.1.1.0-amzn-0 をサポート
  - 上記以外のバージョンもJARをロードすることで利用可能
- **Amazon Athena**
  - Icebergテーブルへのクエリをサポート
  - Glueカタログ上のテーブルのみ、Parquetファイルのみなどの制限あり※
- **AWS Glue**
  - Glue 4.0ではIceberg 1.0.0 をネイティブサポート
  - 上記以外のバージョンもJARをロードすることで利用可能
  - GlueカタログはIcebergテーブルをETLジョブにて登録可能（2023/05現在クローラはIceberg未サポート）
- **Amazon Redshift**
  - 外部表未サポート
- **Amazon Kinesis Data Analytics**
  - JARをロードすることで任意のバージョンを利用可能

※https://docs.aws.amazon.com/athena/latest/ug/querying-iceberg.html

Delta Lake

Delta Lakeとは

- 2019年からDatabricksによって開発されたオープンソースのフォーマット
- Databricks社が開発の主体
  - Databricks社を中心としたコミュニティが活発
- Apache Sparkでのインテグレーションが進んでいる
  - Presto, Trino, Hive, Flinkなどもサポート（read only）
- データファイルはParquet、Avro、ORCで構成される
- 最も早くメジャーバージョンが1になった
- Databricksを利用する場合はDelta Engineを利用できる

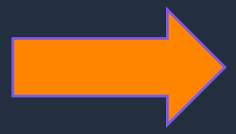
Delta Lakeの構成

ログベースのアプローチ

Transaction Log
Single Commits

(Optional) Partition Directories
Data Files

```
my_table/
  _delta_log/
    00000.json
    00001.json
  date=2019-01-01/
    file-1.parquet
```



Transaction Log
Single Commits

Checkpoint Files
(Optional) Partition Directories
Data Files

```
my_table/
  _delta_log/
    000000.json
    000001.json
    000002.json
       ⋮
    0000010.json
    0000010.checkpoint.parquet
  date=2019-01-01/
    file-1.parquet
  date=2019-01-02/
    file-2.parquet
```

*https://databricks.com/blog/2019/08/21/diving-into-delta-lake-unpacking-the-transaction-log.html

Delta Lakeの特徴

- ACIDトランザクション、タイムトラベル、ストリーミングの機能がそろっている
- Databricksを利用する場合に、Delta Engineやその他のインテグレーションにより高パフォーマンスかつ多機能となる
  - Z-Ordering, 統計情報を利用した読み込みスキップ、Bloom indexes, コンパクション、local SSD caching、同時接続等
  - OSSバージョンはそれよりも低速
- S3, Azure Storage, GCS, HDFS, Oracle Cloud Infrastructure, IBM Cloud Object Storageに最適化されている

AWSアナリティクスサービスの対応状況 （2023/05/16 現在）

- **Amazon EMR**
  - EMRネイティブでのサポートはなし
  - JARをロードすることで任意のバージョンを利用可能
- **Amazon Athena**
  - Athena Engine Version 3 で Delta Lakeテーブルのへのリードクエリをサポート
  - Glueカタログ上のテーブルのみ、タイムトラベル未サポートなどなどの制限あり※
- **AWS Glue**
  - Glue 4.0ではDelta Lake 2.1.0 をネイティブサポート
  - 上記以外のバージョンもJARをロードすることで利用可能
  - Glue カタログおよび Glue クローラでDelta Lakeサポート
- **Amazon Redshift**
  - 外部表でサポート。バージョンは2020/9/24のアナウンス時で0.5.0
- **Amazon Kinesis Data Analytics**
  - JARをロードすることで任意のバージョンを利用可能

※https://docs.aws.amazon.com/athena/latest/ug/delta-lake-tables.html

まとめ

データレイクフォーマットの比較

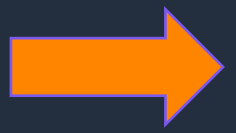
Feature	HUDI	Iceberg	Delta OSS
Transactions (ACID)	Y	Y	Y
Row Level Updates & Deletes	Y	Y	Y
Schema Evolution	Limited	Y (partitions)	Limited
Time Travel	Y	Y	Y
File Format	Parquet	Parquet, Avro,ORC	Parquet
Incremental query support	Y	Y	Y
BackGround Clustering (by key)	Y	N	N
CDC Scenarios	Y	Y	Y
Concurrent writer support	Y	Y	Y
High Throughput Streaming	Y	N	N

サードパーティによる比較

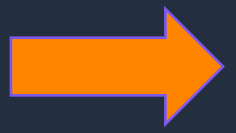
- Comparison of Data Lake Table Formats (Iceberg, Hudi and Delta Lake)
  Updated on Jun 28, 22
- **Delta vs Iceberg vs hudi : Reassessing Performance**
- Transactional Data Lakes — a Comparison of Apache Iceberg, Apache Hudi and Delta Lake **(AWS Data Lab SA による記事)**

データレイクフォーマットの選び方例

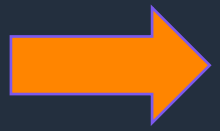
1. 複数のAWSアナリティクスサービスで柔軟にデータを管理する必要がある
2. 高スループットのストリーミングもしくはCDCによる書き込みがある場合



1. 現在Databricksを利用している場合



1. 複雑で巨大なテーブルを最適化して利用する必要がある場合
2. 書き込みより読み込みの比重が大きい場合
3. Amazon Athenaでの書き込みを行いたい場合



Appendix

References

- Hudi
  - https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying
  - https://www.oreilly.com/radar/questioning-the-lambda-architecture/
  - https://www.oreilly.com/content/ubers-case-for-incremental-processing-on-hadoop/
  - https://eng.uber.com/uber-big-data-platform/
  - https://hudi.apache.org/blog/2021/07/21/streaming-data-lake-platform/
  - https://www.onehouse.ai/blog/apache-hudi-vs-delta-lake-vs-apache-iceberg-lakehouse-feature-comparison
- Iceberg
  - https://www.dremio.com/subsurface/fewer-accidental-full-table-scans-brought-to-you-by-apache-icebergs-hidden-partitioning/
  - https://www.slideshare.net/RyanBlue3/iceberg-a-modern-table-format-for-big-data-strata-ny-2018
  - https://www.slideshare.net/RyanBlue3/the-evolution-of-netflixs-s3-data-warehouse-strata-ny-2018
  - https://www.slideshare.net/Hadoop_Summit/iceberg-a-fast-table-format-for-s3-103201179
  - https://www.slideshare.net/PrasannaRajaperumal/hoodie-incremental-processing-on-hadoop
  - https://www.slideshare.net/vinothchandar/hoodie-dataengconf-2017
  - https://www.dremio.com/blog/apache-iceberg-achieves-milestone-1-0-release/
  - https://iceberg.apache.org/releases/

References

- Glue
  - https://aws.amazon.com/blogs/big-data/part-1-integrate-apache-hudi-delta-lake-apache-iceberg-datasets-at-scale-aws-glue-studio-notebook/
  - https://aws.amazon.com/blogs/big-data/build-a-serverless-pipeline-to-analyze-streaming-data-using-aws-glue-apache-hudi-and-amazon-s3/
- Athena
  - https://docs.aws.amazon.com/ja_jp/athena/latest/ug/querying-hudi.html
  - https://docs.aws.amazon.com/ja_jp/athena/latest/ug/querying-iceberg.html
- EMR
  - https://aws.amazon.com/blogs/big-data/new-features-from-apache-hudi-available-in-amazon-emr/
  - https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-iceberg.html
- Redshift
  - https://docs.aws.amazon.com/redshift/latest/dg/c-spectrum-external-tables.html
  - https://aws.amazon.com/jp/about-aws/whats-new/2020/09/amazon-redshift-spectrum-adds-support-for-querying-open-source-apache-hudi-and-delta-lake/

Thank you!