Delta Lake シリーズ

ストリーミング

ストリーミングデータの処理を
Delta Lake で高速化

databricks

このeBookの概要

Databricks の Delta Lake の eBook シリーズは、データを扱う方が Delta Lake のフル機能を理解して利活用するための支援を目的として提供されています。

この eBook「Delta Lake シリーズ：ストリーミング」では、Delta Lake の強力なストリーミング処理機能について詳しく解説します。

学べる内容

Delta Lake によって、静的データのバッチコンピューティングの高速化と同様に、ストリーミングデータのコンピューティングを高速化する方法を理解できます。

databricks

目次

Delta Lake とは

Delta Lake は、データの信頼性を高め、迅速な分析をクラウドのデータレイクにもたらす統合データ管理システムです。既存のデータレイク上で動作し、Apache Spark™ API と完全な互換性があります。

Databricks では、Delta Lake がデータレイクにもたらす信頼性、性能、ライフサイクル管理を実証してきました。私たちのお客様は Delta Lake を活用して、不正形式のデータ取り込みの回避、コンプライアンスに対応するデータ削除、データ取得時のデータ修正など、さまざまな課題を解決しています。

Delta Lake は、高品質データをデータレイクに迅速にもたらし、セキュアでスケーラブルなクラウドサービスで、チームによるデータの利活用を加速させます。

Chapter

01

Delta Lake でストリーミングの「ペインポイント」を解決する

01

Delta Lake でストリーミングの「ペインポイント」を解決する

従来のストリーミングとデータウェアハウスソリューションのペインポイントは、データレイクとデータウェアハウスのペインの 2 つのグループに分けることができます。

データレイクのペインポイント

データレイクでは、膨大な量のデータをファイルシステムに柔軟に保存することができますが、（これに限らず）多くのペインポイントがあります。

- 多くの異種システムからのストリーミングデータの統合は困難です。
- データレイク内のデータを更新することはほぼ不可能であり、ストリーミングデータの多くは変更に合わせて更新する必要があります。これは、財務照合とその後の調整を含むシナリオにおいて特に重要です。
- データレイクのクエリ速度は一般的に非常に遅くなります。
- ストレージやファイルサイズの最適化は非常に難しく、複雑なロジックを必要とすることが多いです。

データウェアハウスのペインポイント

データウェアハウスの力は、データの永続的なパフォーマンスを持つストアを持っていることです。しかし、現代の継続的なアプリケーションを構築するためのペインポイントには、以下のようなものがあります（ただし、これに限定されません）。

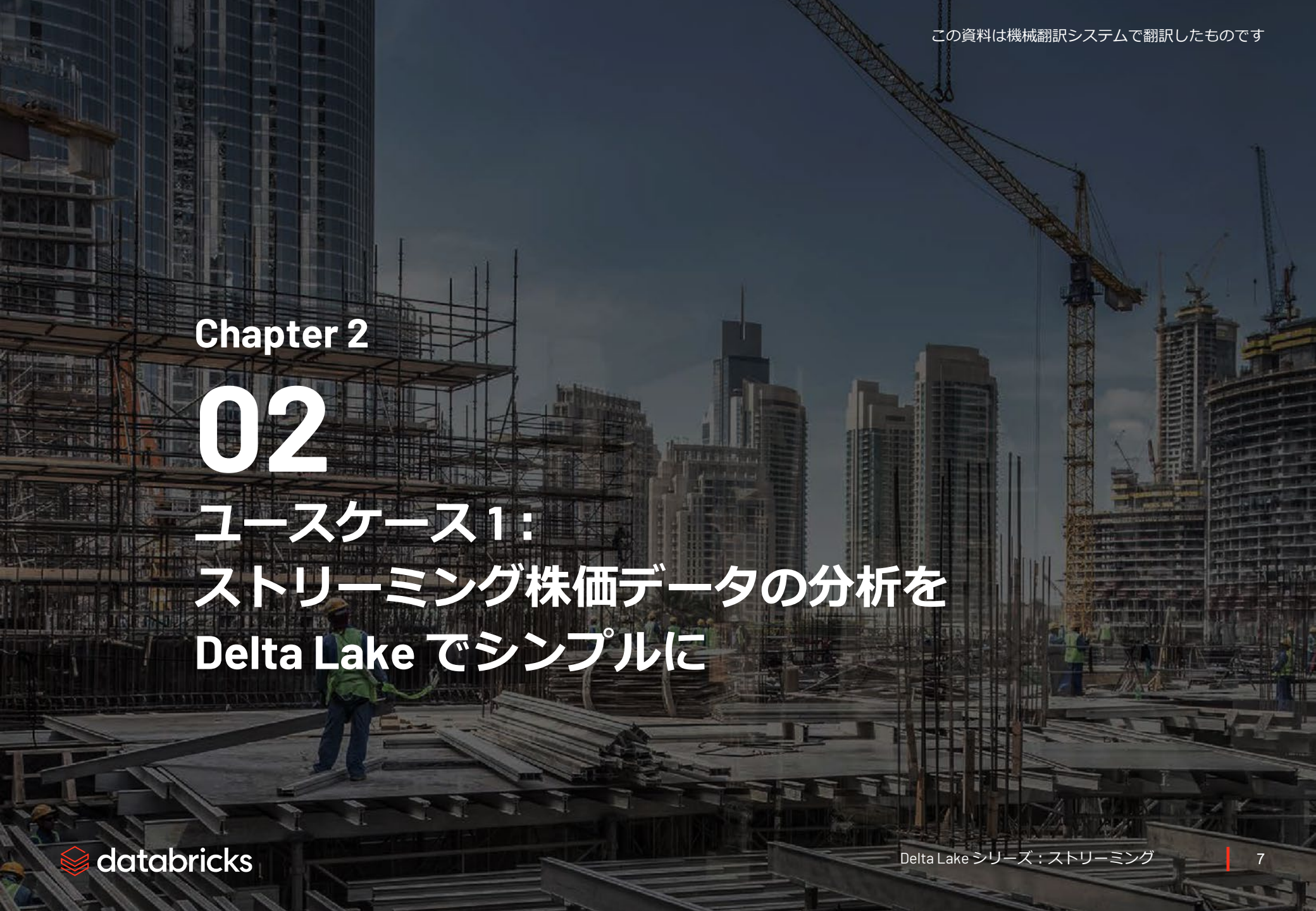
- SQL クエリに制約されている（つまり、機械学習や高度なアナリティクスがない）。
- ストリーミングデータと保存されたデータを一緒にアクセスするのは、可能であるとしても非常に困難です。
- データウェアハウスは、あまりスケール感がありません。
- コンピューティングとストレージを結びつけると、ウェアハウスの使用は非常に高価になります。

Databricks の Delta Lake がいかにしてペインポイントを解決するか

Delta Lake は、クラウドのデータレイクにデータの信頼性とパフォーマンスの最適化をもたらす統合データ管理システムです。簡単に言うと、Delta Lake はデータレイクとデータウェアハウスの利点を Apache Spark™ と組み合わせて、画期的なことができるようにします。

- Delta Lake は、構造化ストリーミングとともに、ストリーミングと履歴データをいっしょに高速に分析することを可能にします。
- ストリーミングビッグデータのソースとデスティネーションとして Delta Lake のテーブルを使用すると、異種のデータソースを容易に統合できます。
- アップサートは Delta Lake のテーブルでサポートされています。
- Delta Lake は ACID に準拠しているため、準拠したデータソリューションを容易に作成できます。
- 機械学習のスコアリングと高度な分析を ETL とクエリに容易に組み込むことができます。
- コンピューティングとストレージを分離し、完全にスケーラブルなソリューションを提供します。

以下の章では、実際のユースケースを紹介します。



Chapter 2

02

ユースケース1：ストリーミング株価データの分析をDelta Lakeでシンプルに

02

ユースケース1：ストリーミング株価データの分析を Delta Lake でシンプルに

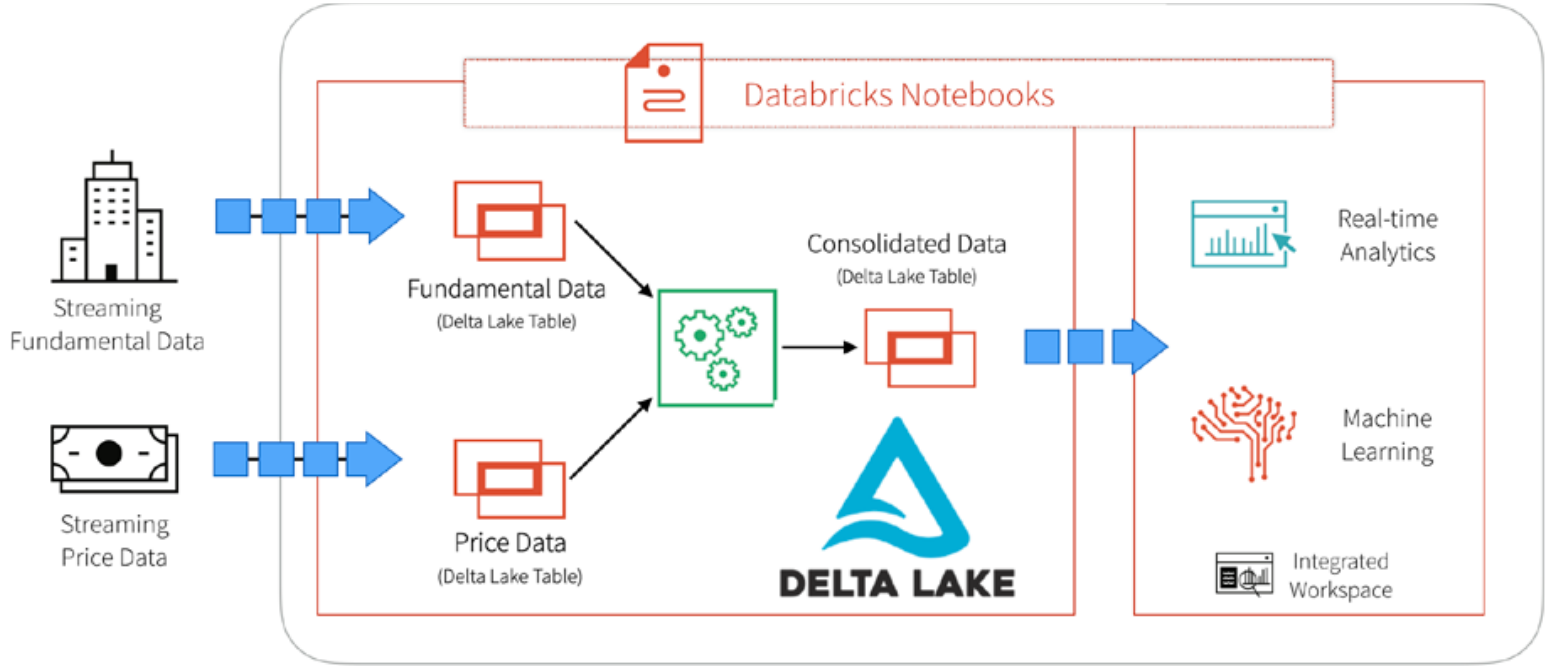
株価データのリアルタイム分析は複雑な作業です。結局のところ、ストリーミングシステムを維持し、レガシーデータとストリーミングデータを並行してトランザクションの一貫性を確保することには多くの課題があります。

ありがたいことに、Delta Lake は、株価データをリアルタイムで分析するためのストリーミングシステムを構築する際の多くの苦手な点を解決するのに役立ちます。このセクションでは、Delta Lake を使って株価データ分析のストリーミングを簡素化する方法を紹介します。

次の図では、この問題を単純化する高レベルのアーキテクチャを見ることができます。まず、2 つの異なるデータセットを 2 つの Delta Lake テーブルにインジェストすることから始めます。 2 つのデータセットは、株価とファンダメンタルズです。

それぞれのテーブルにデータをインジェストした後、ETL プロセスでデータを結合し、下流の分析のために第 3の Delta Lake のテーブルにデータを書き出します。

Delta Lake は、Apache Spark のスケーラビリティ、ストリーミング、高度なアナリティクスへのアクセスと、データウェアハウスのパフォーマンスと ACID コンプライアンスを組み合わせることで、これらの問題を解決します。



ストリーミング株価分析ソリューションを Delta Lake で実装

Delta Lake と Apache Spark がソリューションのほとんどの作業を行っています。Notebook を試してみて、以下のコードサンプルに従ってください。

上の図にあるように、処理するデータセットは 2 つあります。2 つの Delta Lake テーブルを作成するために、Databricks File System（DBFS）の場所に対して `.format('delta')` を指定します。

```
# Create Fundamental Data (Databricks Delta table)
dfBaseFund = spark \\
.read \\
.format('delta') \\
.load('/delta/stocksFundamentals')

# Create Price Data (Databricks Delta table)
dfBasePrice = spark \\
.read \\
.format('delta') \\
.load('/delta/stocksDailyPrices')
```

stockFundamentals と stocksDailyPrices を更新している間に、一連の ETL ジョブを通じてこのデータを連結ビュー socksDailyPricesWFund に集約します。

以下のコードスニペットで、利用可能なデータの開始日と終了日を決定し、その日付範囲の株価とファンダメンタルズデータを DBFS に結合することができます。

```
# Determine start and end date of available data
row = dfBasePrice.agg(
   func.max(dfBasePrice.price_date).alias("maxDate"),
   func.min(dfBasePrice.price_date).alias("minDate")
).collect()[0]
startDate = row["minDate"]
endDate = row["maxDate"]

# Define our date range function
def daterange(start_date, end_date):
   for n in range(int ((end_date - start_date).days)):
      yield start_date + datetime.timedelta(n)

# Define combinePriceAndFund information by date and
def combinePriceAndFund(theDate):
   dfFund = dfBaseFund.where(dfBaseFund.price_date == theDate)
   dfPrice = dfBasePrice.where(
dfBasePrice.price_date == theDate
).drop('price_date')
   # Drop the updated column
   dfPriceWFund = dfPrice.join(dfFund, ['ticker']).drop('updated')
   # Save data to DBFS
    dfPriceWFund
.write
.format('delta')
.mode('append')
.save('/delta/stocksDailyPricesWFund')

# Loop through dates to complete fundamentals + price ETL process
for single_date in daterange(
startDate, (endDate + datetime.timedelta(days=1))
):
   print 'Starting ' + single_date.strftime('%Y-%m-%d')
   start = datetime.datetime.now()
   combinePriceAndFund(single_date)
   end = datetime.datetime.now()
   print (end - start)
```

これで、連結されたファンダメンタルズと株価データのストリームが /delta/stocksDailyPricesWFund の場所にある DBFS にプッシュされるようになりました。DBFS の場所に対して .format("delta") を指定することで、Delta Lake テーブルを作成することができます。

```
dfPriceWithFundamentals = spark
.readStream
.format("delta")
.load("/delta/stocksDailyPricesWFund")

// Create temporary view of the data
dfPriceWithFundamentals.createOrReplaceTempView("priceWithFundamentals")
```

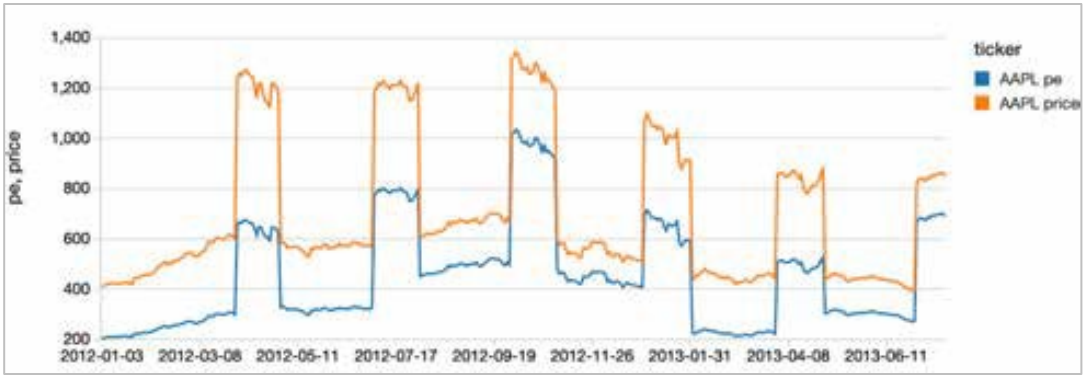
Delta Lake のテーブルを作成したので、リアルタイムで株価収益率を計算できるビューを作成してみましょう（Delta Lake のテーブルを更新しているストリーミングデータがベースにあるため）。

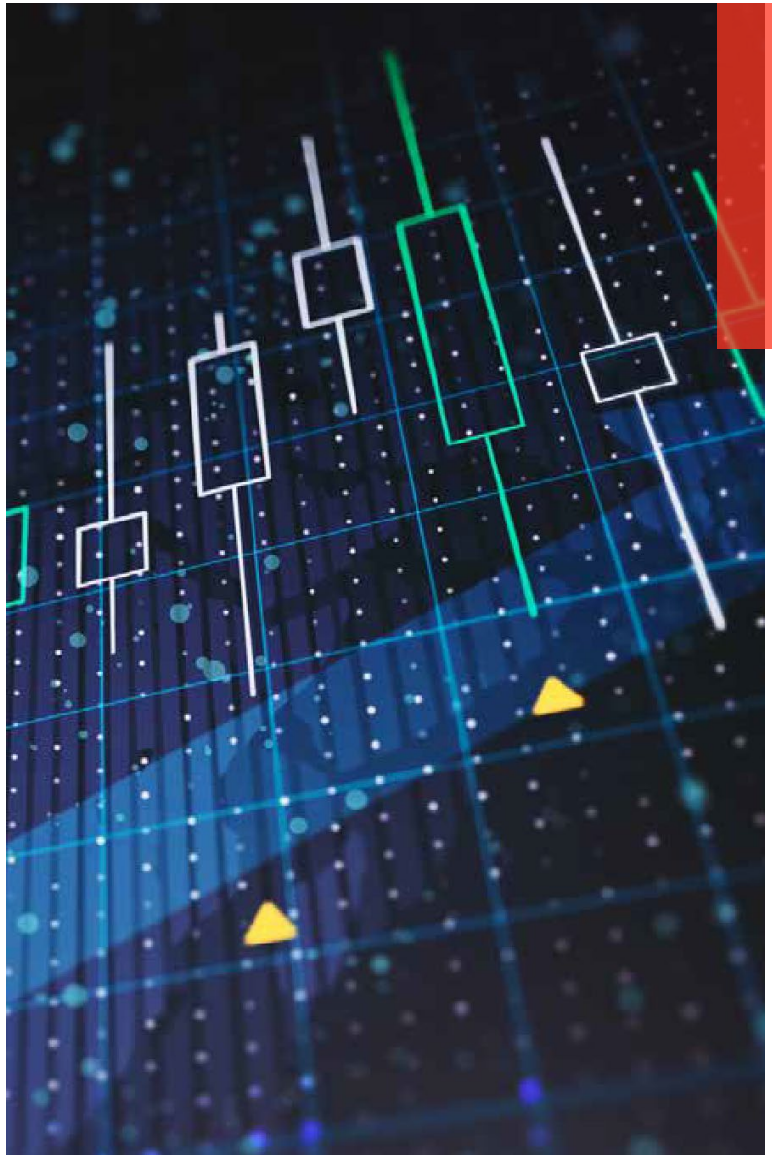
```
%sql
CREATE OR REPLACE TEMPORARY VIEW viewPE AS
select ticker,
     price_date,
     first(close) as price,
     (close/eps_basic_net) as pe
   from priceWithFundamentals
 where eps_basic_net > 0
 group by ticker, price_date, pe
```

ストリーミングされた株価データをリアルタイムで分析

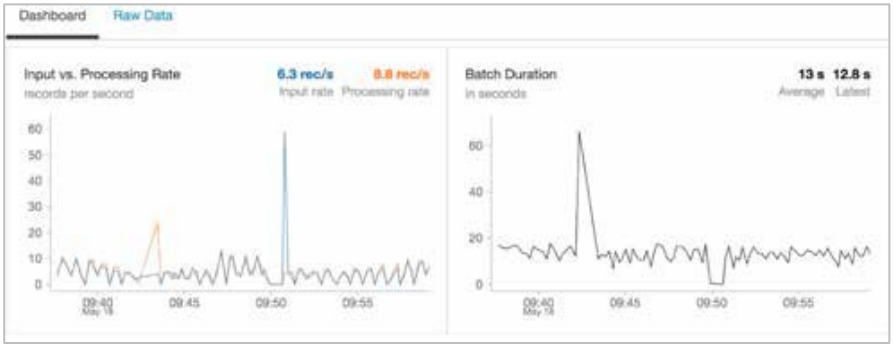
Spark SQL を使ってデータを素早く分析し、Databricks のビュー機能で可視化します。

```
%sql  select *
from viewPE
where ticker == "AAPL"
order by price_date
```

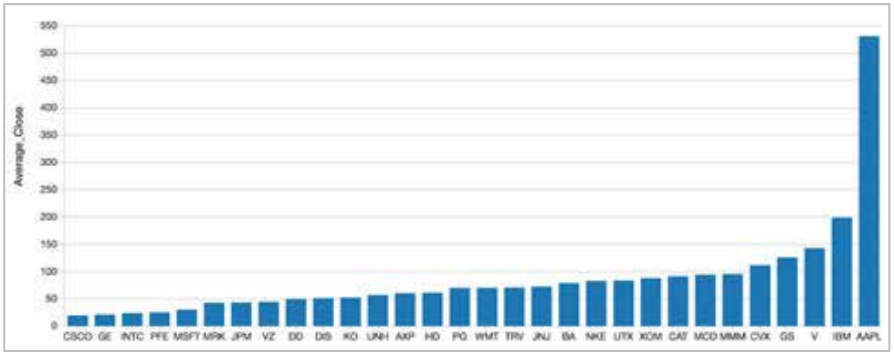




この統合データセットの基礎となるソースは Delta Lake のテーブルなので、このビューはバッチデータだけでなく、次のストリーミングダッシュボードのような新しいデータストリームも表示しています。



Structured Streaming は、Delta Lake のテーブルにデータを書き込むだけでなく、追跡する必要がある特定の数のキー（この場合はティッカーシンボル）の状態を保持しています。



Spark SQL を使用しているため、アグリゲーションクエリを大規模かつリアルタイムで実行できます。

```
%sql
SELECT ticker, AVG(close) as Average_Close
FROM priceWithFundamentals
GROUP BY ticker
ORDER BY Average_Close
```

最後に、Delta Lake を使ってストリーミング株価データの分析を簡素化を実演しました。Spark Structured Streaming と Delta Lake を組み合わせることで、Databricks の統合ワークスペースを利用して、データレイクとデータウェアハウスの両方の利点を持つパフォーマンスの高いスケーラブルなソリューションを構築できます。

Databricks 統合データプラットフォームは、ストリーミングやトランザクションの一貫性に関連したデータエンジニアリングの複雑さを解消し、データエンジニアリングやデータサイエンスチームは株価データのトレンドの理解に集中できます。

Chapter

03

ユースケース 2：Tilting Point 社事例 — Delta Lake へのストリーミングインジェスト

03

ユースケース 2：Tilting Point 社事例 — Delta Lake へのストリーミングインジェスト

Tilting Point（ティルティングポイント）社は、一流の開発スタジオに専門的なリソース、サービス、運用サポートを提供し、高品質なライブゲームを最適化して成功に導く新世代のゲームパートナーです。 Tilting Point 社は、ユーザー獲得ファンドと世界クラスの技術プラットフォームを通じてパフォーマンスマーケティング管理とライブゲームの運営に資金を提供し、開発者が収益性の高い規模を達成できるように支援しています。

Delta Lake を活用することで、 Tilting Point 社は質の高いデータを活用し、ビジネスを改善するためのアナリティクスに利用できるようになりました。このユースケースについて、 Tilting Point 社のエンジニアリング担当副社長である Diego Link 氏は、次のように述べています。

Tilting Point 社のチームは、ゲーム分析のレポート作成のために、毎日、1時間ごとのバッチジョブを実行していました。彼らは、レポートをほぼリアルタイムにして、5〜10 分以内に気づきを得られるようにしたいと考えていました。

また、リアルタイムデータをバンドル＆オファーシステムに提供するために、リアルタイムのプレイヤー行動に基づいてゲーム内の LiveOps の決定を行い、実際に予期せぬ有害な影響を及ぼす可能性のある LiveOps の変更について最新のアラートを提供し、ゲーム運営におけるサービスの中断についてもアラートを提供したいと考えていました。その目標は、ゲーム体験がプレイヤーにとって可能な限り堅牢なものであることを保証することでした。

さらに、GDPR のコンプライアンスを維持するために、暗号化された個人情報（PII）データを別個に保存しなければなりませんでした。

データのフローと、それに伴う課題

Tilting Point 社は、開発者が統合して AWS でホストされているインジェストサーバーにゲームサーバーからデータを送信するための独自のソフトウェア開発キットを持っています。このサービスは、すべての PII データを削除し、生データを Amazon Firehose のエンドポイントに送信します。 その後、Firehose は、JSON 形式のデータを S3 に継続的にダンプします。

生データをクリーンアップして分析に利用できるようにするために、チームは Firehose からメッセージバス（Kafka、Kinesis など）に連続データをプッシュし、Apache Spark の Structured Streaming を使用してデータを連続的に処理し、Delta Lake のテーブルに書き込むことを検討しました。

このアーキテクチャは、データを数秒で処理するという低レイテンシーの要件には理想的に聞こえますが、 Tilting Point 社はインジェストパイプラインにそのような低レイテンシーの要件を持っていませんでした。彼らが求めていたのは、数秒ではなく数分でデータを分析に利用できるようにすることでした。そこで、メッセージバスを排除してアーキテクチャを簡素化し、代わりに S3 を構造化ストリーミングジョブの継続的なソースとして使用することにしました。 しかし、S3 を継続的なソースとして使用する際の重要な課題は、最近変更されたファイルを特定することです。

数分ごとにすべてのファイルをリストアップすることには、2 つの大きな問題があります。

- **遅延発生：**ファイル数の多いディレクトリ内のすべてのファイルをリストアップすると、オーバーヘッドが大きくなり、処理時間が長くなります。
- **コスト増大：**数分ごとに多数のファイルをリストアップすると、S3 のコストが増大します。

ブロブストアをソースとし、Delta Lake のテーブルをシンクとした構造化ストリーミングの活用

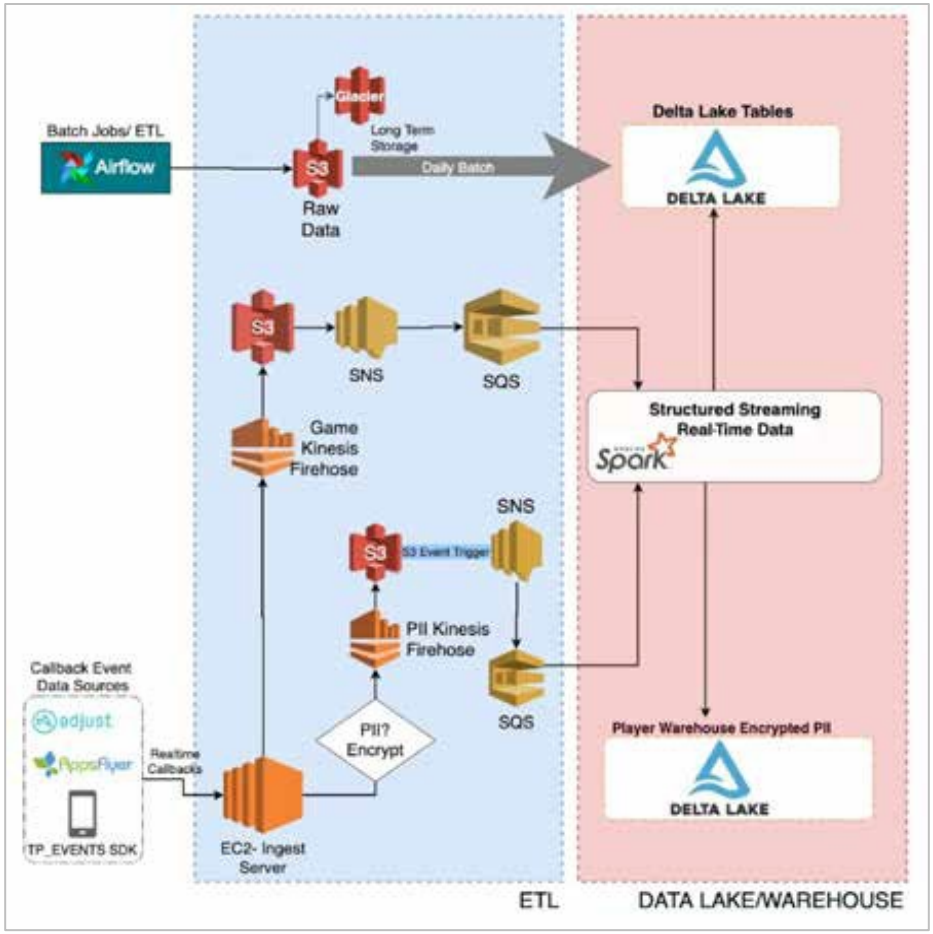
S3 のようなクラウドブロブストレージからデータを連続的にストリーミングするために、 Tilting Point 社は Databricks の S3-SQS ソースを使用しています。S3-SQS ソースは、最近処理されたファイルの状態管理コードを記述することなく、S3 からデータをインクリメンタルにストリーミングする簡単な方法を提供します。

Tilting Point 社のインジェストパイプラインは次のようになっています。

- Amazon S3 のイベント通知を設定して、新規ファイル到着情報を SNS 経由で SQS に送信する。
- Tilting Point 社は、S3に到着した新しいデータを S3-SQS ソースから読み込みます。S3-SQS ソースは、SQS から S3 に到着した新しいファイル名を読み込み、その情報を使って実際のファイル内容を S3 に読み込みます。以下にコード例を示します。

```
spark.readStream \
   .format("s3-sqs") \
   .option("fileFormat", "json") \
   .option("queueUrl", ...) \
   .schema(...) \
   .load()
```

- その後、 Tilting Point 社の構造化ストリーミングジョブは、データをクリーンアップして変換します。 ゲームデータに基づいて、ストリーミングジョブは Spark ストリーミングの foreachBatch API を使用し、30 種類の Delta Lake テーブルに書き込みます。
- ストリーミングジョブは小さなファイルを大量に生成します。これは下流の消費者のパフォーマンスに影響を与えます。そこで、Delta Lake のテーブルからデータを読み込んでいる間、データの消費者が良好なパフォーマンスを発揮できるように、テーブル内の小さなファイルを圧縮し、適切なファイルサイズとして保存するために、毎日最適化ジョブが実行されます。また、 Tilting Point 社では、週に一度、第二ラウンドの圧縮のためのオプティマイズジョブも実行しています。



Delta Lake テーブルへの継続的なデータインジェストを示すアーキテクチャ

上記の Delta Lake のデータ取り込みアーキテクチャは、次のような点で役立ちます。

インクリメンタルローディング
S3-SQS ソースは、S3 内の新しいファイルをインクリメンタルにロードします。 ファイルをリストアップする際のオーバーヘッドをあまりかけずに、新しいファイルを迅速に処理することができます。

明示的なファイル状態管理は不要
最近のファイルを探すための明示的なファイル状態管理は必要ありません。

運用負荷が低い
Firehose ジョブと Structured Streaming ジョブ間のチェックポイントとして S3 を利用しているため、ストリームを停止してデータを再処理する運用負荷は比較的低い。

信頼性の高いデータ取り込み
Delta Lake は楽観的な同時実行制御を使用して、ACID トランザクションの保証を提供しています。これにより、信頼性の高いデータインジェストが可能になります。

ファイルの圧縮
ストリーミングインジェストの大きな問題の 1 つは、テーブルが大量の小さなファイルで終わることで、読み込みパフォーマンスに影響を与える可能性があります。 Delta Lake の前は、圧縮したデータを書き込むために別のテーブルを設定しなければなりませんでした。 Delta Lake では、ACID トランザクションのおかげで、ファイルを圧縮し、同じテーブルに安全にデータを書き戻すことができます。

スナップショット分離
Delta Lake のスナップショット分離により、データがストリーミングジョブによって追加され、圧縮中に変更されている間、インジェストテーブルを下流のコンシューマーに公開できます。

ロールバック
悪い書き込みがあった場合、Delta Lake の Time Travel は、以前のバージョンのテーブルにロールバックするのに役立ちます。

この章では、 Tilting Point 社のユースケースと、Databricks の S3-SQS ソースを使用して Delta Lake テーブルにストリーミングインジェストをどのように行うかを説明し、高品質のデータをアナリティクスに利用するために、運用上のオーバーヘッドをあまりかけずに効率的に行う方法を説明しました。

Chapter

04

ユースケース 3 : ストリーミングビデオサービスの QoS 分析ソリューションの構築方法

04

ユースケース 3 : ストリーミングビデオサービスの QoS 分析ソリューションの構築方法

従来型の有料テレビの低迷が続く中、コンテンツ所有者は、コンテンツのライブラリを収益化するために、D2C（ダイレクトトゥコンシューマー）サブスクリプションや広告付きストリーミングを採用しています。優れたコンテンツを制作し、それを配信者にライセンス供与することをビジネスモデルとしていた企業にとっては、コンテンツを消費者に配信するためのメディアサプライチェーンの構築、無数のデバイスやオペレーティングシステムに対応したアプリのサポート、課金やカスタマーサービスなどの顧客関係機能の実行など、新たな能力が必要とされています。

ほとんどのサービスは月単位で更新されるため、定額制サービスのオペレータは、常に加入者に価値を証明する必要があります。ストリーミングビデオの一般的な品質の問題（バッファリング、遅延、ピクセレーション、ジッター、パケットロス、空白の画面などを含む）は、加入者の解約の増大やビデオのエンゲージメントの低下など、ビジネスに大きな影響を与えます。

ストリーミングを開始すると、途切れが発生する場所が非常に多く、視聴者の体験が損なわれる可能性があることに気付きます。オンプレミスまたはクラウド上のサーバーのソースに問題があるかもしれません。CDN レベル、ISP レベル、または視聴者のホーム ネットワークのいずれかのトランジット中に問題があるかもしれません。n x 104 コンカレント ストリーマーでのブレークは、n x 105 または n x 106 でのブレークとは異なります。チャンネル サーフィンをしたり、アプリをクリックして出入りしたり、異なるデバイスから同時にサインオンしたりするような、実世界のユーザーと、最も冗長なシステムでさえも限界点まで押し上げる能力を再現できるような、リリース前のテストはありません。また、テレビの性質上、最も重要で注目度の高いイベントで最大の視聴者を集めている間は、物事がうまくいかないことがあります。

ソーシャルメディア上で苦情がきた場合、それが一人のユーザーだけの問題なのか、それとも地域的な問題なのか、それとも全国的な問題なのか、どのように見分けますか？全国的な問題だとすると、その問題はすべてのデバイスで起こっているのか、それとも特定のタイプのデバイスだけの問題なのでしょうか（例えば、OEM が古いタイプのデバイスで OS をアップデートしたために、クライアントとの互換性の問題が発生してしまったなど）？

視聴者の体験品質の問題を特定、改善、防止することは、ユーザーの数、ユーザーが実行しているアクションの数、体験中のハンドオフの数（サーバー、CDN、ISP、ホームネットワーク、クライアント）を考慮すると、ビッグデータの問題になります。サービス品質（QoS）は、これらのデータの流れを理解するのに役立ち、何がどこで、なぜうまくいかないのかを理解することができます。最終的には、何がうまくいかないのか、何かが壊れる前にどのように修正するのかを予測分析することができます。

Databricks のサービス品質ソリューションの概要

このソリューションの目的は、QoS システムを改善したいと考えているあらゆるストリーミングビデオプラットフォームのコアを提供することです。このソリューションは、AWS Labs が提供する AWS Streaming Media Analytics ソリューションをベースにしており、その上に Databricks を統合データ分析プラットフォームとして追加し、リアルタイムのインサイトと高度な分析機能を提供しています。

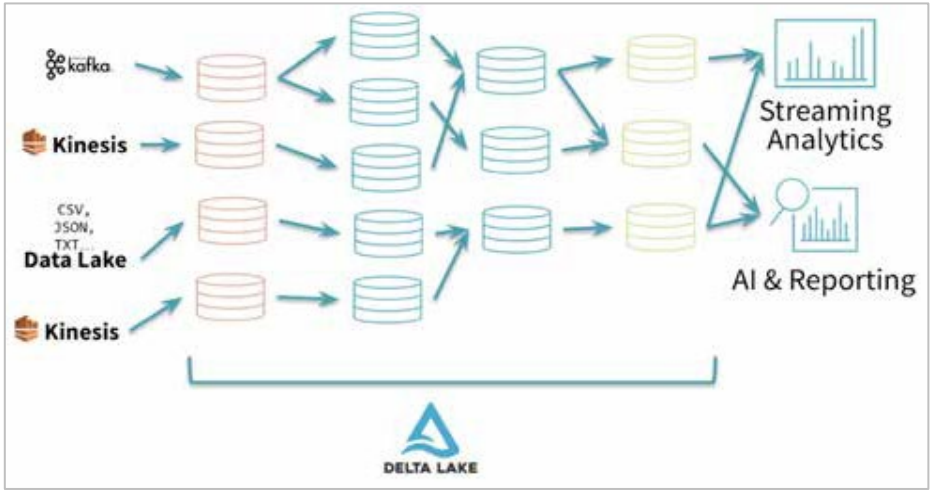
Databricks を使用することで、ストリーミングプラットフォームは、堅牢で信頼性の高いデータパイプラインから供給される最も完全で最新のデータセットを常に活用することで、より迅速なインサイトを得ることができます。コラボレーション環境を使用してデータサイエンスを加速することで、新機能の市場投入までの時間を短縮します。データエンジニアリングとデータサイエンスの両方のための統一されたプラットフォームを持つことで、エンドツーエンドの機械学習ライフサイクルの管理をサポートし、ソフトウェア開発のすべてのサイクルにわたって運用コストを削減します。

ビデオ QoS ソリューションアーキテクチャ

低レイテンシの監視アラートや、ビデオトラフィックのピーク時に必要とされる高度にスケーラブルなインフラストラクチャなどの複雑性を考慮して、簡単に選択したアーキテクチャはデルタアーキテクチャでした。ラムダアーキテクチャやカッパアーキテクチャのような標準的なビッグデータアーキテクチャは、複数のタイプのパイプライン（ストリーミングとバッチ）を維持するために必要な運用上の労力や、統一されたデータエンジニアリングとデータサイエンスのアプローチをサポートしていないという欠点があります。

デルタアーキテクチャは、組織内のすべてのデータペルソナをより生産性の高いものにする次世代のパラダイムです。

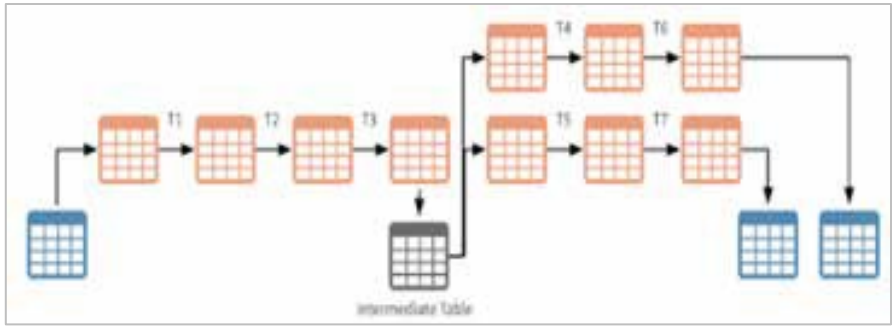
- データエンジニアは、バッチとストリーミングのどちらかを選択することなく、コスト効率の高い方法でデータパイプラインを継続的に開発することができます。
- データアナリストは、ほぼリアルタイムに近い洞察力と BI クエリに対する迅速な回答を得ることができます。
- データサイエンティストは、再現性の高い実験やレポートを容易にするタイムトラベルをサポートすることで、より信頼性の高いデータセットを使用して、より優れた機械学習モデルを開発できます。



データパイプラインのための「マルチホップ」アプローチを使用したデルタアーキテクチャ

デルタアーキテクチャを使用してデータパイプラインを書くことは、データに構造を段階的に追加する多層の「マルチホップ」アプローチのベストプラクティスに従っています。"ブロンズ」テーブルまたは「インジェスト」テーブルは、通常、ネイティブ形式（JSON、CSV、またはtxt）の生データセットであり、「シルバー」テーブルは、レポートやデータサイエンスの準備ができているクリーン化/変換されたデータセットを表し、「ゴールド」テーブルは、最終的なプレゼンテーション層です。

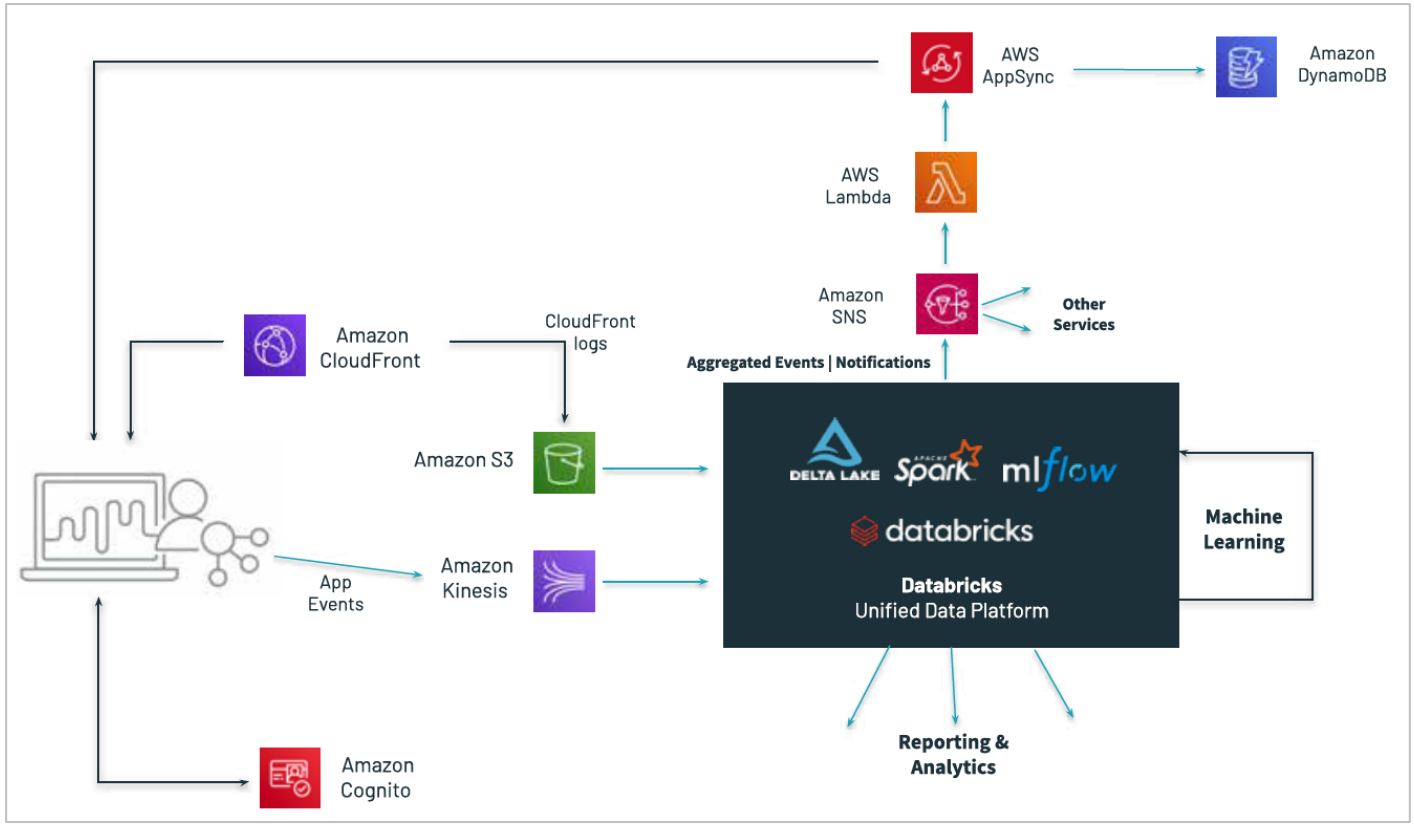
純粋なストリーミングのユースケースでは、Delta Lake の中間テーブルで DataFrames を実現するオプションは、基本的にはレイテンシー／SLA とコストのトレードオフに過ぎません（例としては、リアルタイムのモニタリングアラートと新しいコンテンツに基づくレコメンドシステムの更新など）。



Delta Lake のテーブルで DataFrame をマテリアライズしながら
ストリーミングアーキテクチャを実現

このアプローチにおける「ホップ」の数は、下流の consumers の数、アグリゲーションの複雑さ（例えば、Structured Streaming は、複数のアグリゲーションをチェーン化する周りの特定の制限を強制する）、および運用効率の最大化によって直接影響を受けます。

QoS ソリューションのアーキテクチャは、データ処理のベストプラクティスに焦点を当てており、完全なビデオオンデマンド（VoD）ソリューションではありません。「フロントドア」サービスの Amazon API Gateway のようないくつかの標準的なコンポーネントは、データとアナリティクスに焦点を当てるために高レベルのアーキテクチャから回避されています。



QoS プラットフォームのアーキテクチャ概要

データを分析可能にする

QoS ソリューションに含まれるデータソース（アプリケーションイベントと CDN ログ）はどちらも JSON 形式を使用しており、データ交換には最適ですが、複雑な入れ子構造を表現することはできますが、スケーラブルではなく、データレイク/分析システムのストレージ形式として維持するのは困難です。

組織全体でデータを直接照会できるようにするために、Bronze から Silver へのパイプライン（「データを誰でも利用できるようにする」パイプライン）では、あらゆる未加工形式を Delta Lake に変換し、規制機関が要求するすべての品質チェックやデータのマスキングを含める必要があります。

type	ts	jsonData
stream	2020-04-08T09:53:58.864+0000	{"metricType":"stream","at":105.077308,"rtt":350,"connection_type":"4g","package":"hls","resolution":"640x360","fps":"29.970","avg_bitrate":1330203,"duration":597,"cdn_tracking_id":"pxiujwunmzfiqxa8soettjrjvznfwsovtnh8epu0-6szzy38svi1vg==","user_id":"us-west-2:810f9e88-55e8-4a80-9b83-936446e6df2d","video_id":"bigbuckbunny","playlist_type":"live","timestamp":1586339838864}
stream	2020-04-08T09:54:00.396+0000	{"metricType":"stream","at":415.111827,"rtt":526,"connection_type":"3g","package":"hls","resolution":"640x360","fps":"29.970","avg_bitrate":1350743,"duration":735,"cdn_tracking_id":"jahzaxapw4y-xhrjoescatcytahbjvduwfgcvawqyihu4pagnrsya==","user_id":"us-west-2:cc7af3ef-6cf9-4da1-8274-3212db46be48","video_id":"tearsofsteel","playlist_type":"live","timestamp":1586339640396}
stream	2020-04-08T09:54:01.332+0000	{"metricType":"stream","at":410.095889,"rtt":591,"connection_type":"4g","package":"hls","resolution":"640x360","fps":"29.970","avg_bitrate":1330203,"duration":597,"cdn_tracking_id":"svymfpqarpmuy-zewwi4u3pbdn2sfgcaodm-2rgdpfiihty4zhika==","user_id":"us-west-2:83cb8ebe-4f29-4822-80a2-feb10095a9a2","video_id":"bigbuckbunny","playlist_type":"live","timestamp":1586339641332}
stream	2020-04-08T09:53:53.498+0000	{"metricType":"stream","at":385.057842741,"rtt":523,"connection_type":"not available","package":"hls","resolution":"640x360","fps":"29.970","avg_bitrate":1350743,"duration":735,"cdn_tracking_id":"h3dpitndfpfy9acagudbie3kwck8gknh98fjdu7uqhikgwagc8ga==","user_id":"us-west-2:2f7000b3-1e01-4cba-a0ef-d49b8952833c","video_id":"tearsofsteel","playlist_type":"live","timestamp":1586339633498}
buffer	2020-04-	{"metricType":"buffer","buffer_type":"firstbuffer","at":385.057842741,"rtt":523,"connection_type":"not available","time_milliseconds":386983,"cdn_tracking_id":"h3dpitndfpfy9acagudbie3kwck8gknh98fjdu7uqhikgwagc8ga==","user_id":"us-west-2:2f7000b3-1e01-4cba-

Showing the first 1000 rows.

アプリイベントの Raw 形式

ビデオアプリケーションのイベント

アーキテクチャに基づいて、ビデオ アプリケーションのイベントは Kinesis Streams に直接プッシュされ、スキーマに変更を加えることなく、Delta Lake の append-only テーブルにインジェストされます。

このパターンを使用することで、Kinesis ストリームのスループットをスケーリングすることなく、下流の多数のコンシューマーがストリーミングパラダイムでデータを処理することができます。シンク（最適化をサポート）として Delta Lake テーブルを使用する副作用として、処理ウィンドウのサイズがターゲットテーブル内のファイル数に影響を与えることを心配する必要がありません。

タイムスタンプとメッセージのタイプの両方が JSON イベントから抽出され、データを分割して、消費者が処理したいイベントのタイプを選択できるようにしています。イベントのための単一の Kinesis ストリームを Delta Lake の「イベント」テーブルと組み合わせることで、運用の複雑さを軽減しつつ、ピーク時のスケーリングを容易にしています。

スキーマ

col_name	data_type
browserfamily	string
bytes	string
cdn_source	string
isbot	boolean
origin	string
location	string
logdate	date
logtime	string
osfamily	string
requestid	string
ip	string
resulttype	string
year	int
month	int
day	int
hour	int

詳細情報はシルバーテーブルの JSON から抽出される

CDN ログ

CDN のログは S3 に配信されるので、最も簡単に処理する方法は、追加の設定なしに S3 に到着した新しいデータファイルをインクリメンタルかつ効率的に処理する「Databricks Auto Loader」です。

```
auto_loader_df = spark.readStream.format("cloudFiles") \
  .option("cloudFiles.format", "json") \
  .option("cloudFiles.region", region) \
  .load(input_location)

anonymized_df = auto_loader_df.select('*', ip_
anonymizer('requestip').alias('ip')) \
    .drop('requestip') \
    .withColumn("origin", map_ip_to_location(col('ip')))


anonymized_df.writeStream ¥
  .option('checkpointLocation', checkpoint_location) \
  .format('delta') \
  .table(silver_database + '.cdn_logs')
```

ログには GDPR 規制の下で個人データとみなされる Ips が含まれているため、「あなたのデータを誰もが利用できるようにする」パイプラインには匿名化のステップが含まれていなければなりません。 さまざまなテクニックを使用することができますが、私たちは IPv4 から最後のオクテットを削除し、IPv6 から最後の 80 ビットを削除することにしました。その上で、データセットには、発信国と ISP プロバイダの情報も含まれており、これは後でネットワークオペレーションセンターでローカライズのために使用されます。

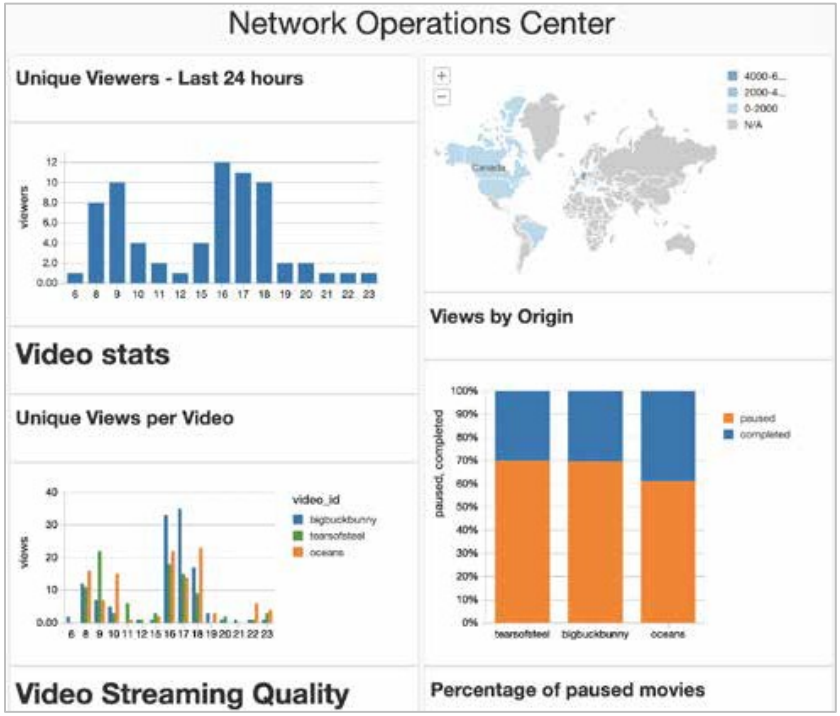
ダッシュボードの作成／仮想ネットワーク運用センター

ストリーミング企業は、ネットワークパフォーマンスとユーザーエクスペリエンスを可能な限りリアルタイムに監視し、個人レベルまで追跡し、セグメントレベルで抽象化して、ジオ、デバイス、ネットワーク、現在および過去の視聴行動によって定義された新しいセグメントを簡単に定義する必要があります。

ストリーミング企業にとっては、通信事業者のネットワークオペレーションセンター（NOC）のコンセプトを採用し、ユーザーのストリーミングエクスペリエンスの健全性をマクロレベルで監視し、あらゆる問題を早期に指摘して対応することを意味しています。最も基本的なことですが、NOC には、ユーザーの現在の体験をパフォーマンスのベースラインと比較するダッシュボードがあり、製品チームがサービスの異常を迅速かつ簡単に特定して対処できるようになっています。

QoS ソリューションには、Databricks ダッシュボードが組み込まれています。BI ツールを簡単に接続して、より複雑なビジュアライゼーションを構築することもできますが、顧客からのフィードバックによると、ほとんどの場合、ビジネスユーザーにインサイトを提示するには、組み込みのダッシュボードが最も早い方法です。

NOC のための集約テーブルは、基本的にはデルタアーキテクチャのゴールド層となります。 CDN のログとアプリケーションのイベントの組み合わせです。

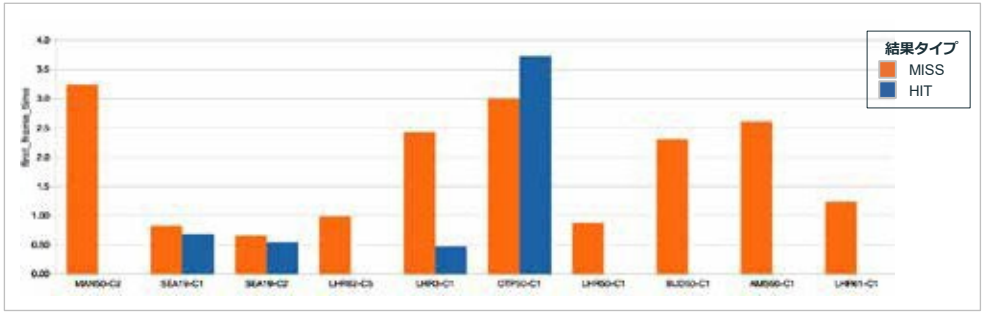


ネットワークオペレーションセンターのダッシュボードの例

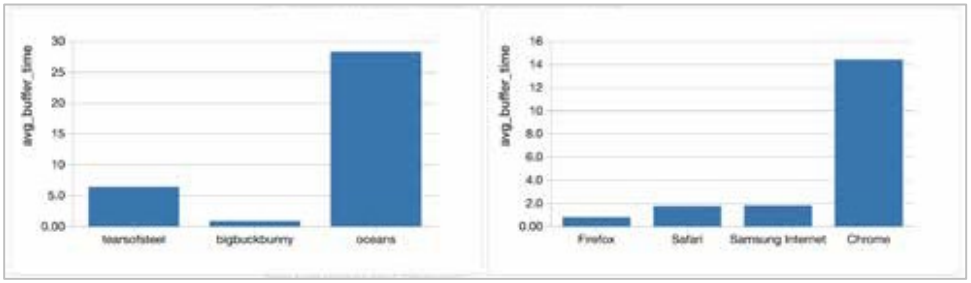
ダッシュボードは、SQL クエリまたは Python/R 変換の結果を視覚的にパッケージする方法に過ぎません。各 Notebook は複数のダッシュボードをサポートしているため、異なる要件を持つ複数のエンドユーザーの場合は、コードを複製する必要はありません。ボーナスとして、更新を Databricks ジョブとしてスケジュールすることもできます。

SQL クエリの結果の可視化

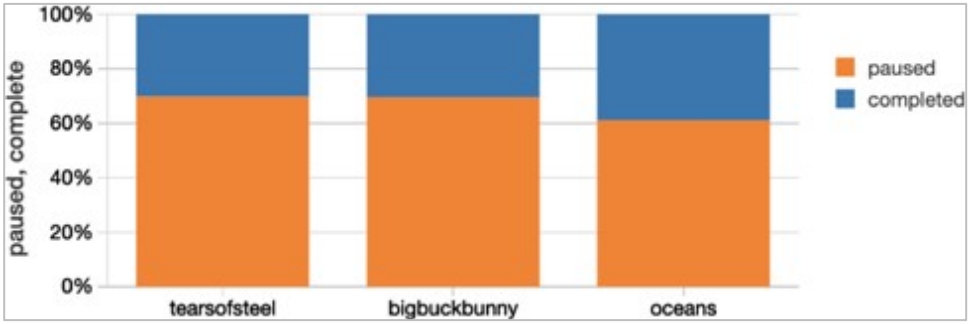
ビデオのロード時間（最初のフレームまでの時間）は、CDNの個々のロケーション（この場合は AWS CloudFront Edge ノード）のパフォーマンスをよりよく理解することができ、このKPIを改善するための戦略に直接影響を与えます。 高レベルのバッファリングの理由を理解していないと、それがもたらすビデオの質の低さは、加入者の解約率に大きな影響を与えます。

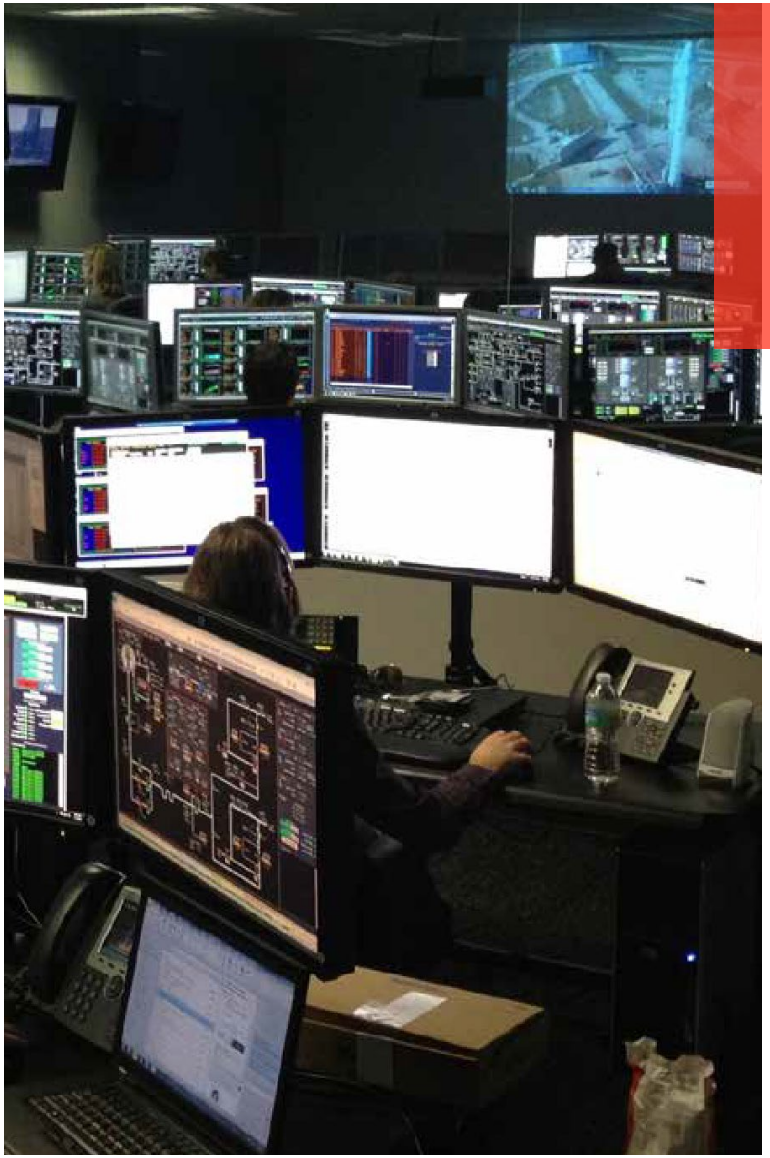


そのうえ、広告主は、視聴者のエンゲージメントを減らすために責任のある広告にお金を費やすことを望んでいない - 彼らは上に余分なバッファリングを追加するので、広告事業の利益は通常、あまりにも影響を受けています。このような状況では、ビデオレベルだけでなく、ブラウザやアプリケーションの種類・バージョンまで分析できるようにするためには、アプリケーション側から可能な限り多くの情報を収集することが非常に重要です。



コンテンツ側では、アプリケーションのイベントは、ユーザーの行動や全体的な体験の質に関する有益な情報を提供することができます。ビデオを一時停止した人のうち、実際にそのエピソード／ビデオの視聴を終えた人の数は？停止の原因は何か。コンテンツの品質か、配信の問題か？もちろん、すべてのソース（ユーザーの行動、CDN／ISP のパフォーマンス）をリンクさせて、ユーザープロファイルを作成するだけでなく、解約を予測することで、さらなる分析を行うことができます。



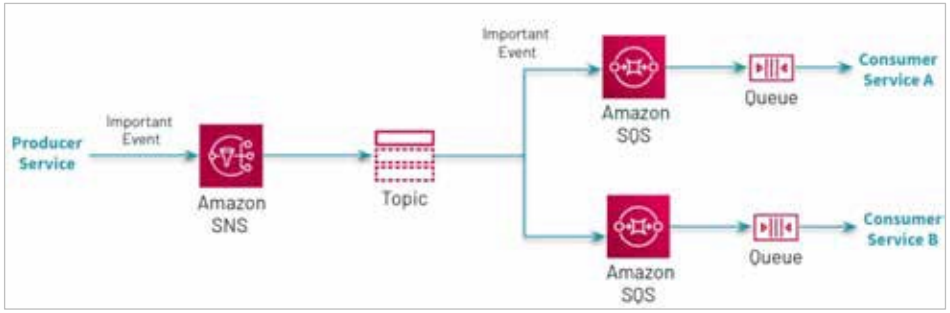


ほぼリアルタイムのアラートの作成

何百万人もの同時使用ユーザーからのビデオストリーミングで生成されるデータの速度、量、種類を扱う場合、ダッシュボードが複雑になると、NOC の人間のオペレータが現時点で最も重要なデータに集中し、根本原因の問題をゼロインすることが困難になることがあります。このソリューションを使用すると、パフォーマンスが特定のしきい値を超えたときに自動アラートを簡単に設定することができ、ネットワークの人間のオペレータに役立つだけでなく、ラムダ関数を介して自動修復プロトコルを設定することができます。例えば、以下のようになります。

- CDN がベースラインよりもはるかに高いレイテンシを持っている場合（例えば、ベースラインの平均に対して 10% 以上のレイテンシを持っている場合）、CDN トラフィックの自動シフトを開始
- 再生エラーを報告するクライアントの数が閾値（例：5%）を超えた場合は、特定のデバイスのクライアントに問題がある可能性が高いことを製品チームに警告
- 特定の ISP の視聴者が平均以上のバッファリングやピクセレーションの問題を抱えている場合は、フロントラインの顧客担当者に対応や問題を軽減する方法（ストリームの品質を低く設定するなど）について注意を喚起

技術的な観点からリアルタイムアラートを生成するには、リアルタイムにデータを処理できるストリーミングエンジンと、プッシュ通知を行うパブリッシュ配信サービスが必要です。



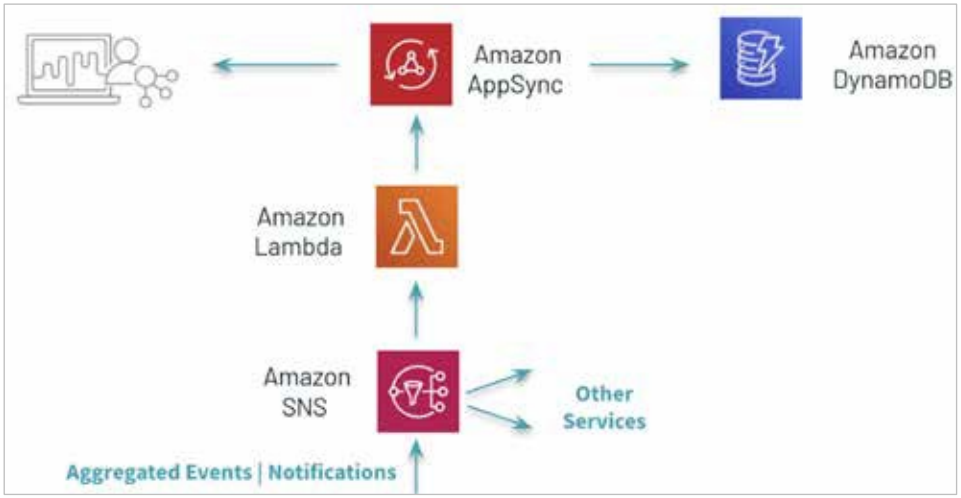
Amazon SNS と Amazon SQS を使ったマイクロサービスの統合

QoS ソリューションは、Amazon SNS とその統合機能を利用してマイクロサービスを統合するためのAWSのベストプラクティスを実装しており、Amazon Lambda（Webアプリケーションの更新については後述）や他のコンシューマー向けの Amazon SQS を利用することができます。書き込みのカスタマイズオプションは、ルールベースのエンジンに基づいてメール通知を送信するパイプラインの記述（例えば、個々のタイプのアプリのエラーの割合を一定期間にわたって検証するなど）を容易にしてくれます。

```
   def send_error_notification(row):

      sns_client = boto3.client('sns', region)

      error_message = 'Number of errors for the App has exceeded
the threshold {}'.format(row['percentage'])

      response = sns_client.publish(
         TopicArn=,
         Message= error_message,
         Subject=,
         MessageStructure='string')

      # Structured Streaming Job

      getKinesisStream("player_events")\
         .selectExpr("type", "app_type")\
         .groupBy("app_type")\
         .apply(calculate_error_percentage)\
         .where("percentage > {}".format(threshold))\
         .writeStream\
         .foreach(send_error_notification)\
         .start()
```

AWS の SNS を利用してメール通知を送信する

QoS ソリューションへの拡張の可能性に焦点を当てながら、顧客ベースの重要なユースケースをいくつか検討してきました。



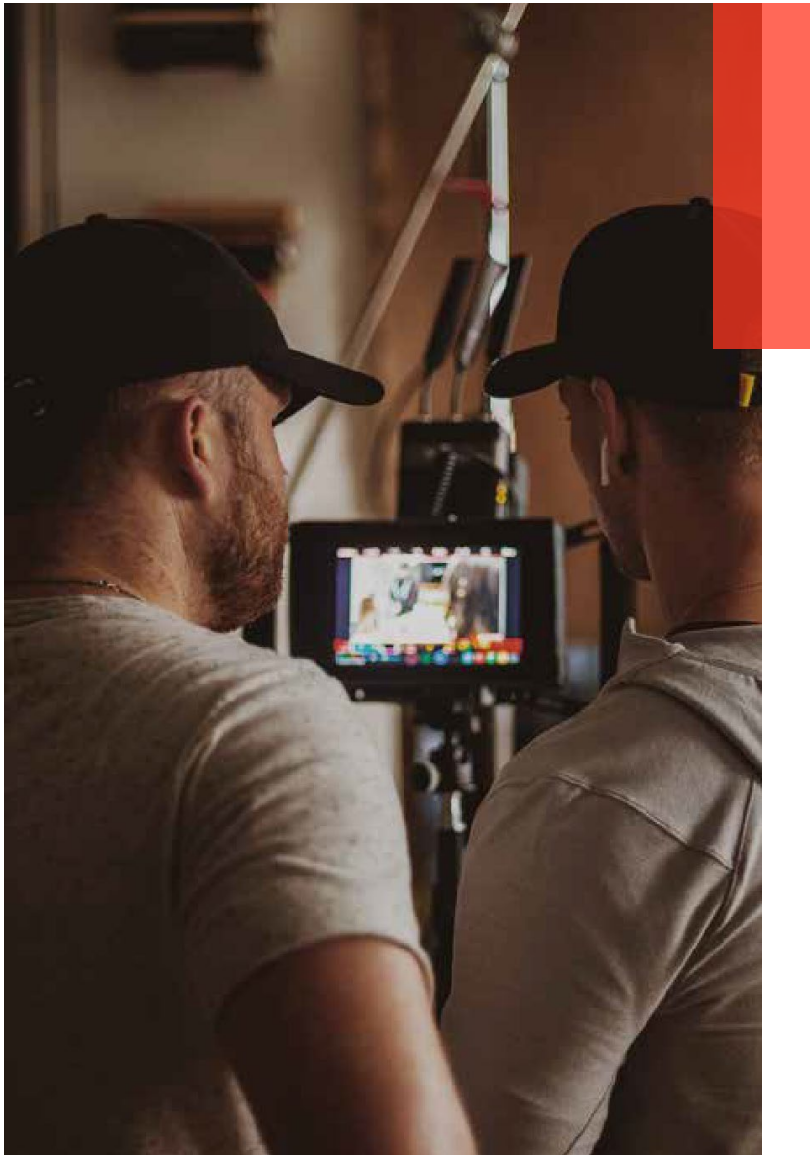
リアルタイムの集計結果でアプリケーションを更新する

QoS ソリューションは、構造化ストリーミングと Amazon SNS の類似したアプローチを適用しており、AWS SQS を使用して追加のコンシューマーを接続できるように、すべての値を更新しています。これは、膨大な量のイベントを強化・分析しなければならない場合によく見られるパターンで、データを一度事前に集計し、各サービス（消費者）が下流で独自の判断を下せるようにしています。

次のステップ：機械学習

手動で過去のデータを意味づけすることは重要ですが、非常に時間がかかります。将来的に自動化された判断ができるようにするには、機械学習アルゴリズムを統合しなければなりません。

統合データプラットフォームとして、Databricks は、Hyperopt、Horvod、AutoML をサポートする機械学習用ランタイムや、エンドツーエンドの機械学習ライフサイクル管理ツール MLflow との統合などの機能を通じて、データサイエンティストによる優れたデータサイエンス製品の構築を支援しています。



失敗予測のポイントと改善策

D2C ストリーマーがより多くのユーザーに届くようになると、瞬間的なサービス損失のコストが増加します。ML は、問題が発生する可能性のある場所を予測し、問題が発生する前に対処することで、オペレータが報告から予防へと移行するのに役立ちます（例えば、同時視聴者が急増した場合には、自動的に容量の大きい CDN に切り替えることができます）。

顧客離脱

サブスクリプションサービスを成長させる上で重要なのは、加入者を維持することです。個人レベルでサービスの品質を理解することで、解約や顧客生涯価値モデルの変数として QoS を追加することができます。さらに、プロアクティブなメッセージングをテストしたり、オファーを節約したりするために、ビデオの品質に問題があった人のために顧客コホートを作成することもできます。

Databricks のストリーミングビデオ QoS ソリューションを使う

ストリーミングビデオのエクスペリエンスに一貫した品質を提供することは、プラットフォーム上でさまざまなエンターテイメントオプションを持つ気まぐれな視聴者を維持するために、極めて重要です。 このソリューションでは、ほとんどのストリーミングビデオプラットフォーム環境で、この QoS リアルタイムストリーミング分析ソリューションを埋め込むためのクイックスタートを実現しました。

- あらゆる規模の視聴者に対応
- 配信ワークフローの主要部分で品質パフォーマンスの問題を迅速に検知
- 新しい自動アラートの作成や、データサイエンティストによる予測分析や機械学習のテストやロールアウトなど、オーディエンスやニーズに合わせて簡単にカスタマイズできる柔軟性とモジュール性

開始するには、Databricks ストリーミングビデオ QoS ソリューションの Notebook をダウンロードしてください。バッチデータとストリーミングデータを単一のシステムに統合する方法の詳細は、Delta Architecture の Web セミナーをご覧ください。

次のステップ

この eBook では、Delta Lake とその機能が性能を向上させる仕組みについて解説しました。このシリーズの他の eBook では、Delta Lake のリソースを詳しくご紹介しています。

この eBook の後続シリーズ

- Delta Lake シリーズ：基礎と性能
- Delta Lake シリーズ：機能
- Delta Lake シリーズ：レイクハウス
- Delta Lake シリーズ：顧客ユースケース

Delta Lake をさらに詳しく

- 技術トークシリーズ：Delta Lake 基本
- 技術トークシリーズ：Delta Lake 詳細
- Databricks の Web サイト
- Databricks の無料トライアル
- Web セミナー：Delta Lake オープンソースの信頼性

databricks