Amazon Redshift のクエリのベストプラクティス

AWS 規範ガイダンス

AWS 規範ガイダンス: Amazon Redshift のクエリのベストプラクティス

Table of Contents

Amazon Redshift のクエリのベストプラクティス

Amazon Web Services (AWS)、Ethan Stark

2024 年 6 月 (ドキュメント履歴)

概要

このガイドでは、Amazon Redshift でクエリとテーブルのパフォーマンスを最適化するための推奨事項とベストプラクティスについて説明します。Amazon Redshift では、標準 SQL を使用して、データウェアハウスとデータレイク全体でペタバイトの構造化データおよび半構造化データをクエリできます。また、このガイドでは、Amazon Redshift データウェアハウスのコアアーキテクチャコンポーネントの概要についても説明します。テーブルプロパティ、クラスター設定、クエリ構造などのクエリパフォーマンス要因についての理解とともに、この知識は、Amazon Redshift データウェアハウスの効率的かつ効果的なテーブルとクエリの設計に役立ちます。

対象者

このガイドは、Amazon Redshift でテーブルとクエリを設計または使用するデータエンジニア、データアーキテクト、データアナリストを対象としています。

目的

このガイドは、お客様と組織が以下の目標を達成することに役立ちます。

- データストレージと取得オペレーションを最適化するためのテーブルを設計する
- 最適なパフォーマンスとコスト削減のためのクエリを設計する
- Amazon Redshift Spectrum のパフォーマンスを最適化して、Amazon Simple Storage Service (Amazon S3) のファイルから直接データをクエリする

Amazon Redshift データウェアハウスのアーキテクチャコンポーネント

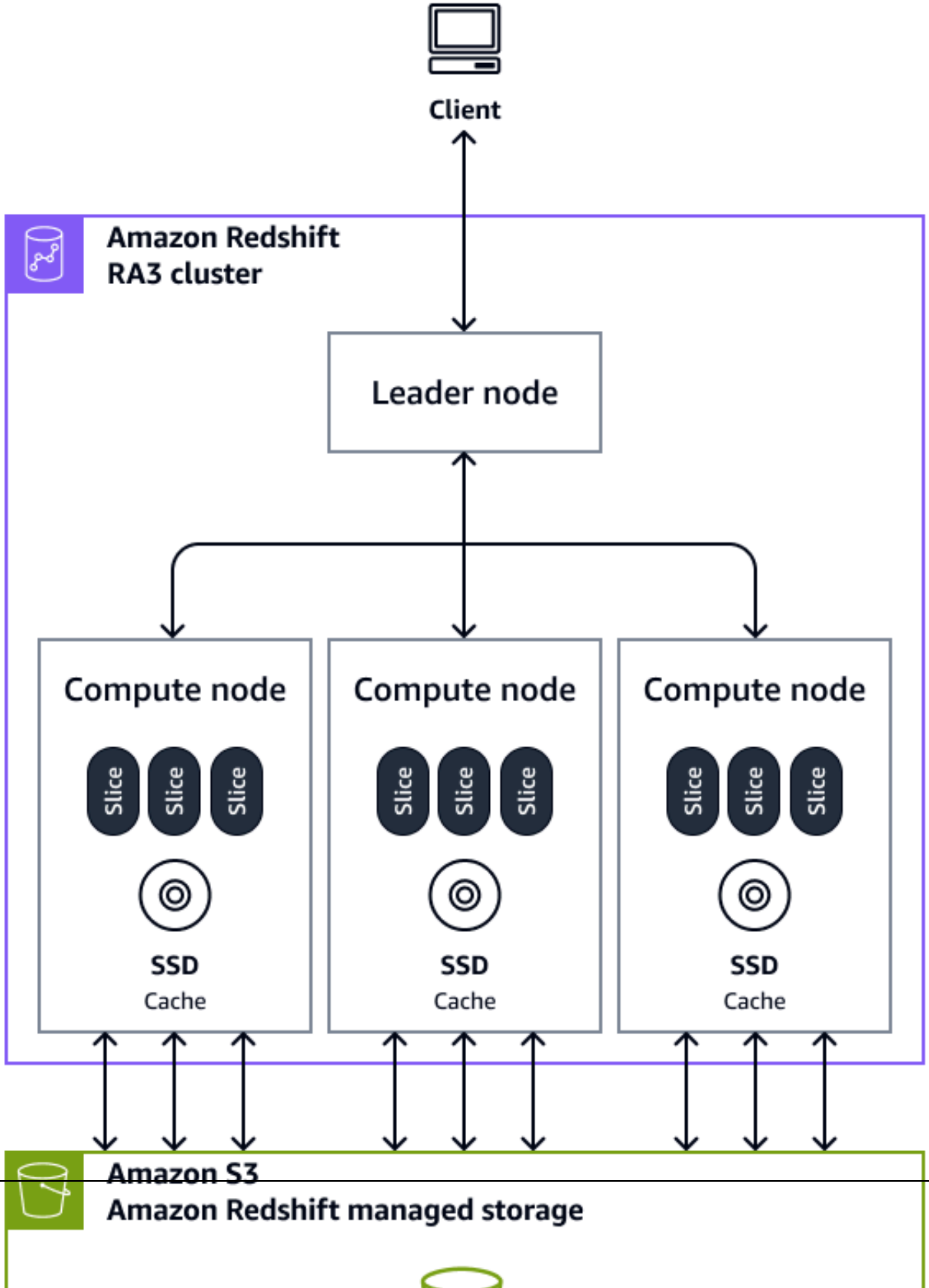
Amazon Redshift データウェアハウスのコアアーキテクチャコンポーネントの基本事項について理解しておくことをお勧めします。この知識は、最適なパフォーマンスを実現するためにクエリとテーブルを設計する方法をよりよく理解するのに役立ちます。

Amazon Redshift のデータウェアハウスは、次のコアアーキテクチャコンポーネントで構成されます。

- クラスター – 1 つ以上のコンピューティングノードで構成されるクラスターは、Amazon Redshift データウェアハウスのコアインフラストラクチャコンポーネントです。コンピューティングノードは外部アプリケーションに対して透過的ですが、クライアントアプリケーションはリーダーノードとのみ直接やり取りします。一般的なクラスターには、2 つ以上のコンピューティングノードがあります。コンピューティングノードは、リーダーノードを介して調整されます。
- リーダーノード – リーダーノードは、クライアントプログラムとすべてのコンピューティングノードの通信を管理します。また、リーダーノードは、クエリがクラスターに送信されるたびにクエリを実行するためのプランも準備します。プランの準備が整うと、リーダーノードでコードをコンパイルし、コンパイル済みのコードをコンピューティングノードに配布してから、データの一部分を各コンピューティングノードに割り当てて、クエリ結果を処理します。
- コンピューティングノード – コンピューティングノードでクエリを実行します。リーダーノードは、クエリ実行プランの個々の要素にコードをコンパイルし、コードを個々のコンピューティングノードに割り当てます。コンピューティングノードは、コンパイル済みのコードを実行し、最終的な集計のために中間結果をリーダーノードに返送します。各コンピューティングノードには、専用の CPU、メモリ、接続されているディスクストレージがあります ワークロードが増えるに従って、ノードの数を増やすかノードの種類をアップグレードして、または両方を行って、クラスターのコンピューティング能力とストレージ容量を拡張できます。
- ノードスライス – コンピューティングノードはスライスと呼ばれる単位に分割されます。コンピューティングノードの各スライスにノードのメモリとディスク容量の一部が割り当てられ、ノードに割り当てられたワークロードの一部分を処理します。スライスは、並列処理を行って操作を完了します。データは、特定のテーブルの分散スタイルと分散キーに基づいて、スライス間で分散されます。データを均等に分散することで、Amazon Redshift がワークロードをスライスに均等に割り当てられるようになり、並列処理の利点を最大化できます。コンピューティングノードあたりのスライス数は、ノードのタイプに基づいて決定されます。詳細については、Amazon Redshift ドキュメントの「Amazon Redshift のクラスターとノード」を参照してください。

- 超並列処理 (MPP) – Amazon Redshift は MPP アーキテクチャを使用して、複雑なクエリや膨大な量のデータであっても、データを迅速に処理します。複数のコンピューティングノードが一部のデータに同じクエリコードを実行して、並列処理を最大化します。
- クライアントアプリケーション – Amazon Redshift は、さまざまなデータロード、抽出、変換、およびロード (ETL)、ビジネスインテリジェンス (BI) レポート、データマイニング、そして分析ツールと統合されています。すべてのクライアントアプリケーションは、リーダーノードを介してのみクラスターと通信します。

次の図は、Amazon Redshift データウェアハウスのアーキテクチャコンポーネントがどのように連携してクエリを高速化するかを示しています。


Client
Amazon Redshift
RA3 cluster
Leader node
Compute node
Slice
Slice
Slice
SSD
Cache
Compute node
Slice
Slice
Slice
SSD
Cache
Compute node
Slice
Slice
Slice
SSD
Cache
Amazon S3
Amazon Redshift managed storage

クエリのライフサイクルには、次の 7 つのステージがあります。

1. クエリの受信と解析:
   - リーダーノードはクエリを受け取り、SQL を解析します。
   - パーサーは、元のクエリの論理構造を表す初期クエリツリーを生成します。
   - Amazon Redshift は、このクエリツリーをクエリオプティマイザに入力します。
2. クエリの最適化:
   - オプティマイザを使用してクエリを評価し、必要なときはクエリを書き換えて効率性を最大限に高めます。
   - このプロセスにより、関連するクエリが複数作成されて、単一のクエリが置き換えられることがあります。
3. クエリプランの生成:
   - オプティマイザによって、実行用のクエリプラン (または必要に応じて複数のプラン) が生成されます。
   - クエリプランは、結合の種類、結合の順序、集計方法、データ分散要件などの実行オプションを指定します。
4. 実行エンジンの変換:
   - 実行エンジンは、クエリプランをステップ、セグメント、ストリームに変換します。
     - ステップ – クエリの実行中に必要な個々のオペレーションを表します。コンピューティングノードはステップを組み合わせることによってクエリ、結合、または他のデータベース操作を実行できます。
     - セグメント – 1 つのプロセスで実行できる複数のステップを組み合わせます。セグメントは、コンピューティングノードのスライスによって実行可能な最小コンパイルユニットです (スライスは、Amazon Redshift の並列処理単位です)。
     - ストリーム – 使用可能なコンピューティングノードのスライスに分配されたセグメントのコレクション。
   - 実行エンジンは、ステップ、セグメント、ストリームに基づいてコンパイルされたコードを生成します。コンパイルされたコードの実行は解釈されたコードよりも速く、使用するコンピューティングキャパシティも少なくなります。
   - リーダーノードは、コンパイルされたコードをコンピューティングノードにブロードキャストします。
5. 同時実行:
   - このステップは、ストリームごとに 1 回実行されます。

- コンピューティングノードのスライスは、クエリセグメントを並列的に実行します。
- このプロセスの間は、Amazon Redshift でネットワーク通信、メモリ使用量、ディスク管理を最適化し、クエリプランのステップから次のステップに中間結果を渡します。
- この最適化は、クエリ実行の高速化に役立ちます。

6. ストリーミング処理:
   - このステップは、ストリームごとに 1 回実行されます。
   - エンジンでストリームごとに実行可能セグメントを作成し、効率的な並列処理を実現します。
7. 最終的なソートと集計:
   - リーダーノードは、クエリに必要な最終的なソートまたは集計に対処します。
   - 完了したら、リーダーノードからクライアントに結果が返されます。

アーキテクチャコンポーネントについては、Amazon Redshift ドキュメントの「データウェアハウスシステムのアーキテクチャ」を参照してください。

Amazon Redshift でのクエリのパフォーマンス要因

いくつかの要因がクエリパフォーマンスに影響を与える可能性があります。データ、クラスター、データベース操作の次の側面はすべて、クエリ処理の速度に影響を与えます。

- テーブルプロパティ
  - ソートキー (Amazon Redshift Advisor)
  - データ圧縮 (自動)
  - データのディストリビューション (自動)
  - テーブルのメンテナンス (自動)
- クラスターの設定
  - ノードタイプ
  - ノードサイズ、ノード数、スライス数
  - ワークロード管理 (自動)
  - ショートクエリアクセラレーション (自動)
- SQL クエリ
  - クエリ構造
  - コードコンパイル

テーブルプロパティ

Amazon Redshift テーブルは、Amazon Redshift にデータを保存するための基本的な単位であり、各テーブルには、その動作とアクセシビリティを決定する一連のプロパティがあります。これらのプロパティには、ソート、分散スタイル、圧縮エンコーディングなどが含まれます。これらのプロパティを理解することは、Amazon Redshift テーブルのパフォーマンス、セキュリティ、コスト効率を最適化するために不可欠です。

ソートキー

Amazon Redshift は、テーブルのソートキーに応じたソート順で、データをディスクに保存します。クエリオプティマイザとクエリプロセッサは、コンピューティングノード内のデータの保存場所に関する情報を使用して、スキャンする必要があるブロックの数を減らします。これにより、処理するデータの量が減り、クエリ速度が大幅に向上します。ソートキーを使用して、WHERE 句のフィル

ターを容易にすることをお勧めします。詳細については、Amazon Redshift のドキュメントの「ソートキーの使用」を参照してください。

データ圧縮

データ圧縮により必要なストレージが減るので、ディスク I/O が減少し、クエリのパフォーマンスが向上します。クエリを実行すると、圧縮されたデータがメモリに読み込まれ、クエリの実行時に圧縮解除されます。メモリにロードするデータを少なくできるので、Amazon Redshift は、より多くのメモリをデータ分析のために割り当てられるようになります。列指向ストレージでシーケンシャルに格納される類似のデータに対して、Amazon Redshift では、列指向のデータ型と明示的に結び付けられた適応型圧縮エンコーディングを利用できます。テーブルの列でデータの圧縮を有効にする最良の方法は、テーブルにデータをロードする際に、Amazon Redshift の AUTO オプションを使用して、最適な圧縮エンコーディングが適用されるようにすることです。自動データ圧縮の使用の詳細については、Amazon Redshift ドキュメントの「自動圧縮ありでテーブルをロードする」を参照してください。

データのディストリビューション

Amazon Redshift は、テーブルの分散スタイルに応じて、データをコンピューティングノードに保存します。クエリを実行すると、クエリオプティマイザが結合と集計を実行するための必要に応じて、データをコンピューティングノードに再分散します。テーブルに適した分散スタイルを選択すると、結合を実行する前にデータを必要な場所に配置しておくことによって、再分散ステップの影響を最小限に抑えることができます。最も一般的な結合を容易にするために、分散キーを使用することをお勧めします。詳細については、Amazon Redshift のドキュメントの「データディストリビューションスタイルの操作」を参照してください。

テーブルのメンテナンス

Amazon Redshift はほとんどのワークロードで業界標準のパフォーマンスを提供しますが、Amazon Redshift クラスターを正常に実行し続けるにはメンテナンスが必要です。データを更新および削除すると、バキューム処理が必要なデッド行が作成され、追加順序がソートキーと一致しない場合は、追加専用テーブルも再ソートする必要があります。

Vacuum

Amazon Redshift でのバキューム処理プロセスは、Amazon Redshift クラスターのヘルスとメンテナンスに不可欠です。また、クエリのパフォーマンスにも影響します。削除や更新によって古いデータにフラグが付きますが、実際に削除されたわけではないため、バキューム処理を使用して、前の UPDATE および DELETE 操作で削除対象としてマークされたテーブル行によって占有されたディス

ク領域を再利用する必要があります。Amazon Redshift は、バックグラウンドでテーブルを自動的にソートし、VACUUM DELETE オペレーションを実行できます。

ロードまたは一連の増分更新の後にテーブルをクリーンアップするには、データベース全体または個々のテーブルに対して VACUUM コマンドを実行することもできます。テーブルにソートキーがあり、テーブルのロードが挿入時にソートするように最適化されていない場合は、バキュームを使用してデータを再ソートする必要があります (これはパフォーマンスにとって重要です)。詳細については、Amazon Redshift ドキュメントの「Vacuuming tables」を参照してください。

分析

ANALYZE 操作によって、Amazon Redshift データベース内のテーブルの統計メタデータを更新します。統計を最新状態に保つことで、クエリプランナーが最適なプランを選択できるようになるため、クエリのパフォーマンスが向上します。Amazon Redshift はデータベースを継続的にモニタリングし、バックグラウンドで自動的に分析オペレーションを実行します。システムパフォーマンスへの影響を最小限にするために、ANALYZE 操作はワークロードが軽い期間に自動で実行されます。ANALYZE を明示的に実行することを選択した場合は、以下を実行する必要があります。

- クエリを実行する前に ANALYZE コマンドを実行します。
- 定期的なロードまたは更新サイクルが終わるたびに、データベースで ANALYZE コマンドを定期的に実行します。
- 作成した新しいテーブルと大幅に変更された既存のテーブルまたは列で ANALYZE コマンドを実行します。
- クエリでの使用と変更傾向に基づき、異なるタイプのテーブルおよび列に対し、異なるスケジュールで ANALYZE 操作を実行することを考慮します。
- 時間とクラスターリソースを節約するには、ANALYZE コマンドを実行するときに PREDICATE COLUMNS 句を使用します。

クラスターの設定

クラスターは、データの実際の保存と処理を実行するノードのコレクションです。以下を実現するには、Amazon Redshift クラスターを正しい方法で設定してください。

- 高いスケーラビリティと同時実行性
- Amazon Redshift の効率的な使用
- パフォーマンスの向上

- コストの削減

ノードタイプ

Amazon Redshift クラスターは、複数のノードタイプ (RA3、DC2、DS2) のいずれかを使用できます。各ノードタイプは様々なサイズを提供し、クラスターを適切に拡張することができるように制限します。ノードサイズによって、クラスター内の各ノードのストレージ容量、メモリ、CPU、および料金が決まります。コストとパフォーマンスの最適化は、適切なノードタイプとサイズを選択することから始まります。ノードタイプの詳細については、Amazon Redshift ドキュメントの「Amazon Redshift クラスターの概要」を参照してください。

ノードサイズ、ノード数、スライス数

コンピューティングノードはスライスに分割されています。ノードが多いということは、プロセッサとスライスが多いことを意味するため、クエリの各部分をスライス間で同時実行することによりプロセスをすばやく処理することが可能になります。ただし、ノード数が多いほど、コストも高くなります。つまり、システムに適したコストとパフォーマンスのバランスを見つける必要があります。Amazon Redshift クラスターアーキテクチャの詳細については、Amazon Redshift ドキュメントの「データウェアハウスシステムのアーキテクチャ」を参照してください。

ワークロード管理

Amazon Redshift のワークロード管理 (WLM) により、ユーザーはワークロードのキューに優先順位を付けて柔軟に管理することが可能になります。これにより、実行速度が高く処理時間の短いクエリが、処理時間の長いクエリの後に滞らないようにできます。自動 WLM は、機械学習 (ML) アルゴリズムを使用してクエリをプロファイリングし、クエリの同時実行とメモリ割り当てを管理しながら、適切なリソースを使用して適切なキューに配置します。WLM の詳細については、Amazon Redshift ドキュメントの「ワークロード管理の実装」を参照してください。

ショートクエリアクセラレーション

ショートクエリアクセラレーション (SQA) は、実行時間が短いクエリを、実行時間が長いクエリよりも優先します。SQA では実行時間が短いクエリを専用領域で実行します。このため SQA クエリは、実行時間が長いクエリをキューで待機するよう強制されません。SQA は、実行時間が短く、ユーザー定義のキュー内にあるクエリのみを優先します。SQA を使用すると、実行時間が短いクエリの実行開始が早くなり、ユーザーへの結果表示も早くなります。SQA を有効にすると、実行時間の短いクエリ専用の WLM キューを減らす、またはなくすことができます。さらに、長時間実行されるクエリは、WLM キュー内のスロットに対して競合する必要はありません。つまり、より

少ないクエリスロットを使用するように WLM キューを設定できます。同時実行数が減るとクエリのスループットが向上し、大部分のワークロードに関するシステム全体のパフォーマンスも向上します。SQA の詳細については、Amazon Redshift ドキュメントの「「ショートクエリアクセラレーションを使用する」を参照してください。

SQL クエリ

データベースクエリは、データベースからのデータのリクエストです。リクエストは SQL を使用して Amazon Redshift クラスターに送信される必要があります。Amazon Redshift は、Java Database Connectivity (JDBC) および Open Database Connectivity (ODBC) を介して接続する SQL クライアントツールをサポートします。JDBC または ODBC ドライバーをサポートするほとんどの SQL クライアントツールを使用できます。

クエリ構造

クエリの書き込み方法は、そのパフォーマンスに大きな影響を与えます。クエリを作成して処理し、ニーズに合わせて必要な最小限のデータを返すことをお勧めします。クエリの構造方法の詳細については、このガイドの「Amazon Redshift クエリの設計のベストプラクティス」セクションを参照してください。

コードコンパイル

Amazon Redshift は、クエリ実行プランごとに最適化されたコードを生成してコンパイルします。コンパイル済みコードは、インタプリタを使用してオーバーヘッドを排除するため、高速で実行されます。コンパイルされたコードのパフォーマンス上の利点を維持しながら、新しいクエリのレイテンシーを最小限に抑えるために、Amazon Redshift はコンポジションと呼ばれる手法を使用します。コンポジションは、既存のロジックの軽量な配置を生成して新しいクエリをすぐに処理し、同時に高度に最適化されたクエリ固有のコードをバックグラウンドでコンパイルします。これにより、クエリ実行のクリティカルパスからコンパイルが削除されます。つまり、新しいクエリはより高速に開始され、後続の実行と一貫したパフォーマンスを提供します。

また、Amazon Redshift はサーバーレスコンパイルサービスを使用して、Amazon Redshift クラスターのコンピューティングリソースを超えてクエリコンパイルをスケールします。コンパイルされたコードセグメントは、クラスター上のローカルキャッシュと、クラスターの再起動後も保持される実質的に無制限のリモートキャッシュの両方にキャッシュされます。同じクエリをそれ以降に実行すると、コンパイルフェーズをスキップできるため、高速になります。スケーラブルなコンパイルサービスを使用することで、Amazon Redshift はコードを並行してコンパイルし、一貫して高速なパフォーマンスを提供します。

Amazon Redshift テーブル設計のベストプラクティス

このセクションでは、データベーステーブルの設計に関するベストプラクティスの概要を説明します。クエリのパフォーマンスと効率を最適化するには、以下のベストプラクティスに従うことをお勧めします。

ソートキーの仕組みを理解する

Amazon Redshift は、ソートキーに応じたソート順でデータをディスクに保存します。Amazon Redshift クエリオプティマイザは、最適なクエリプランを決定する際にソート順を使用します。ソートキーを効果的に使用するには、以下を実行することをお勧めします。

- テーブルは可能な限りソートされたままにします。
- VACUUM ソートを使用して最適なパフォーマンスを復元します。
- ソートキー列を圧縮しないでください。
- ソートキーが圧縮され、sortkey1_skew 比率が著しく高い場合は、ソートキーで圧縮を有効にせずにテーブルを再作成します。
- ソートキー列に関数を適用しないでください。例えば、次のクエリでは、trans_dt : TIMESTAMPTZ ソートキー列を DATE にキャストする場合、そのソートキー列は使用されません。

```
select order_id, order_amt
from sales
where trans_dt::date = '2021-01-08'::date
```

- ソートキーの順序で INSERT オペレーションを実行します。
- 可能な場合は、GROUP BY 句でソートキーを使用します。

クエリ調整のヒント

クエリを調整するには、以下を実行することをお勧めします。

- 最適な効果を得るには、常に複合ソートキーをカーディナリティの低いものから高いものの順に並べます。
- 複合ソートキーの先頭キーが比較的一意である (つまり、カーディナリティが高い) 場合は、ソートキーに列を追加しないでください。列を追加してもクエリのパフォーマンスにはほとんど影響せず、メンテナンスコストが増えていきます。

ソートキーの有効性を評価する

クエリを最適化するには、クエリの有効性を評価できるようにする必要があります。SVL_QUERY_SUMMARY ビューを使用して、クエリの実行についての全般的な情報を確認することをお勧めします。このビューでは、IS_RRSCAN 属性を使用して、EXPLAIN プランのステップで範囲制限付きスキャンを使用しているかどうかを判断できます。また、rows_pre_filter 属性を使用して、ソートキーの選択性を決定することもできます。

v_my_last_query_summary という GitHub の管理者ビューを使用することもできます。ビューには、最後に実行されたクエリの情報が表示されます。

次のステートメントは、クエリの実行に関する一般的な情報を検索する方法を示しています。

```
select lpad(' ',stm+seg+step) || label as label,
       rows,
       bytes,
       is_diskbased,
       is_rrscan,
       rows_pre_filter
from svl_query_summary
where query = pg_last_query_id()
order by stm, seg, step;
```

前述のクエリは、次のサンプル出力を返します。

label	rows	bytes	is_diskbased	is_rrscan	rows_pre_filter
scan tbl=163860 name=orders	1500000	24000000	f	f	1500000
project	1500000	0	f	f	0
project	1500000	0	f	f	0
hash tbl=968	1500000	24000000	f	f	0
scan tbl=163852 name=lineitem	6001215	144029160	f	t	6001215
project	6001215	0	f	f	0
project	6001215	0	f	f	0
hjoin tbl=968	6001215	0	f	f	0
project	6001215	0	f	f	0
project	6001215	0	f	f	0

テーブルについて理解する

テーブルの重要なプロパティを理解することが重要です。テーブルの詳細については、以下を実行してください。

- PG_TABLE_DEF を使用して、テーブル列に関する情報を表示します。
- SVV_TABLE_INFO を使用すると、データ分散スキュー、キー分散スキュー、テーブルサイズ、統計情報など、テーブルに関するより包括的な情報を表示することができます。

適切なテーブルの分散スタイルを選択する

クエリを実行すると、必要に応じて結合と集計を実行するために、クエリオプティマイザによって行がコンピューティングノードに再分散されます。テーブル分散スタイルの選択は、クエリを実行する前にデータを必要な場所に配置しておくことによって、再分散ステップの影響を最小限に抑えるために行われます。

適切なテーブル分散スタイルを選択するには、次のアプローチをお勧めします。

- 同じノード内の行をコロケーションすることで、クエリ実行プランでのブロードキャストと再分散を回避します。例えば、DISTKEY を選択すると、ファクトテーブルと 1 次元テーブルを共通の列に分散できます。フィルタリングされたデータセットのサイズに基づいて最大ディメンションを選択します。結合で使用されている行のみが分散される必要があるため、テーブルのサイズではなく、フィルタリング後のデータセットのサイズを考慮します。
- 分散キーが作成された列に歪みがないことを確認します。歪みがあると、1 つのコンピューティングノードが他のコンピューティングノードよりも重い負荷をかける可能性があります。歪みがあることに気付いた場合は、分散キー列の変更を検討してください。列の値が均一に分散されているか、カーディナリティ値が高い場合、列を分散キーの候補と見なすことができます。
- 結合条件で使用されるテーブルが小さい (1 GB 未満) 場合は、分散スタイル ALL を検討してください。
- 分散キーを圧縮することはできますが、ソートキー列 (特にソートキーの最初の列) を圧縮しないようにする必要があります。

> **Note**
>
> 自動テーブル最適化を使用する場合、テーブルの分散スタイルを選択する必要はありません。詳細については、Amazon Redshift ドキュメントの「Working with automatic table optimization」を参照してください。Amazon Redshift が適切なディストリビューションスタイルを選択できるようにするには、ディストリビューションスタイルに AUTO を指定します。

Amazon Redshift クエリの設計のベストプラクティス

このセクションでは、クエリの設計に関するベストプラクティスの概要を説明します。最適なクエリのパフォーマンスと効率性を実現するには、このセクションのベストプラクティスに従うことをお勧めします。

SELECT * FROM ステートメントを使用しない

SELECT * FROM ステートメントは使用しないことをお勧めします。代わりに、分析する列を必ず一覧表示してください。これにより、クエリの実行時間が短縮され、Amazon Redshift Spectrum クエリのスキャンコストが削減されます。

避けるべき内容の例

```
select *
from sales;
```

ベストプラクティスの例

```
select sales_date, sales_amt
from sales;
```

クエリの問題を特定する

STL_ALERT_EVENT_LOG ビューを確認し、クエリで発生する可能性のある問題を特定して修正することをお勧めします。

クエリの概要情報を取得する

クエリの概要情報を取得するには、SVL_QUERY_SUMMARY ビューと SVL_QUERY_REPORT ビューを使用することをお勧めします。この情報を使用して、クエリを最適化できます。

クロス結合を避ける

やむを得ない場合を除き、クロス結合を使用することは避けてください。クロス結合は、2 つのテーブルの直積集合を算出する結合条件のない結合です。クロス結合は通常ネステッドループ結合として実行されますが、これは使用可能な結合タイプで最も低速な結合です。

避けるべき内容の例

```
select c.c_name,
          n.n_name
from tpch.customer c,
        tpch.nation n;
```

ベストプラクティスの例

```
select c.c_name,
           n.n_name
from tpch.customer c,
join tpch.nation n
  on n.n_nationkey = c.c_nationkey;
```

クエリ述語で関数を使用しない

クエリ述語では関数を使用しないことをお勧めします。通常、関数は各行に処理オーバーヘッドを追加し、クエリの実行全体を遅くするため、クエリ述語で関数を使用すると、パフォーマンスに悪影響を及ぼす可能性があります。

避けるべき内容の例

```
select sum(o_totalprice)
from tpch.orders
where datepart(year, o_orderdate) = 1992;
```

ベストプラクティスの例

```
select sum(o_totalprice)
from tpch.orders
where o_orderdate between '1992-01-01' and '1992-12-31';
```

不要なキャスト変換を避ける

データ型のキャストには時間とリソースがかかり、クエリの実行が遅くなるため、クエリで不要なキャスト変換を使用しないことをお勧めします。

避けるべき内容の例

```
select sum(o_totalprice)
from tpch.orders
where o_ordertime::date = '1992-01-01';
```

ベストプラクティスの例

```
select sum(o_totalprice)
from tpch.orders
where o_ordertime between '1992-01-01 00:00:00' and '1992-12-31 23:59:59';
```

複雑な集計には CASE 式を使用する

同じテーブルから複数回選択するのではなく、CASE 式を使用して複雑な集計を実行することをお勧めします。

避けるべき内容の例

```
select sum(sales_amt) as us_sales
from sales
where country = 'US';

select sum(sales_amt) as ca_sales
from sales
where country = 'CA';
```

ベストプラクティスの例

```
select sum(case when country = 'US' then sales_amt end) as us_sales,
           sum(case when country = 'CA' then sales_amt end) as ca_sales
from sales;
```

サブクエリを使用する

クエリのテーブル 1 つが述語条件のみで使用されている場合には、サブクエリを使用することをお勧めします。この場合、サブクエリはわずかな数の行を返します (ほぼ 200 行未満)。

避けるべき内容の例

サブクエリが 200 行未満を返す場合:

```
select sum(order_amt) as total_sales
from sales
where region_key IN
        (select region_key
         from regions
         where state = 'CA');
```

ベストプラクティスの例

サブクエリが 200 行以上を返す場合:

```
select sum(o.order_amt) as total_sales
from sales o
join regions r
  on r.region_key = o.region_key
  and r.state = 'CA';
```

述語を使用する

述語を使用してデータセットをできるだけ制限することをお勧めします。述語は SQL で使用され、クエリで返されるデータをフィルタリングおよび制限します。述語で条件を指定することで、指定された条件に基づいてクエリ結果に含める必要がある行を指定できます。これにより、関心のあるデータのみを取得でき、クエリの効率と精度が向上します。詳細については、Amazon Redshift ドキュメントの「Conditions」を参照してください。

述語を追加して結合を使用するテーブルをフィルタリングする

述語が同じフィルタを適用する場合でも、述語を追加して結合に関与するテーブルをフィルタリングすることをお勧めします。述語を使用して SQL で結合を使用するテーブルをフィルタリングすると、処理する必要があるデータの量と中間結果セットのサイズを減らして、クエリのパフォーマンスを向上させることができます。WHERE 句で結合オペレーションの条件を指定することで、クエリ実行エンジンは、結合前に条件に一致しない行を削除できます。これにより、結果セットが小さくなり、クエリの実行が高速化されます。

避けるべき内容の例

```
select p.product_name, sum(o.order_amt)
from sales o
```

```
join product p
   on r.product_key = o.product_key
where o.order_date > '2022-01-01';
```

ベストプラクティスの例

```
select p.product_name, sum(o.order_amt)
from sales o
join product p
  on p.product_key = o.product_key
  and p.added_date > '2022-01-01'
where o.order_date > '2022-01-01';
```

述語には最も安価な演算子を使用する

述語では、できる限りコストのかからない演算子を使用します。比較条件演算子は LIKE 演算子よりも優先されます。LIKE 演算子は、SIMILAR TO 演算子または POSIX 演算子よりも優先されます。

GROUP BY 句でソートキーを使用する

GROUP BY 句でソートキーを使用すると、クエリプランナーがより効率的な集計を使用できます。GROUP BY リストにソートキー列のみが含まれており、そのうち 1 つが分散キーでもある場合、クエリは 1 フェーズ集計の対象となる可能性があります。GROUP BY リストのソートキー列には、1 番目のソートキーの後に、ソートキー順序で使用する他のソートキーが含まれている必要があります。

マテリアライズドビューの利点

可能であれば、複雑なコードをマテリアライズドビューに置き換えてクエリを書き換えます。これにより、クエリのパフォーマンスが大幅に向上します。詳細については、Amazon Redshift のドキュメントの「Amazon Redshift でのマテリアライズドビューの作成」を参照してください。

GROUP BY 句と ORDER BY 句の列に注意する

ORDER BY 句と GROUP BY 句の両方を使用する場合は、列を GROUP BY 句と ORDER BY 句の両方に同じ順序で配置してください。GROUP BY では、データを暗黙的にソートする必要があります。ORDER BY 句が異なる場合は、データを 2 回ソートする必要があります。

避けるべき内容の例

```
select a, b, c, sum(d)
from a_table
group by b, c, a
order by a, b, c
```

ベストプラクティスの例

```
select a, b, c, sum(d)
from a_table
group by a, b, c
order by a, b, c
```

Amazon Redshift Spectrum を使用する際のベストプラクティス

このセクションでは、Amazon Redshift Spectrum を使用するためのベストプラクティスの概要について説明します。Redshift Spectrum を使用する際に最適なパフォーマンスを実現するには、以下のベストプラクティスに従うことをお勧めします。

- ファイルタイプが Redshift Spectrum クエリのパフォーマンスに重大な影響を与えることを考慮してください。パフォーマンスを向上させるには、ORC や Parquet などの列エンコードされたファイルを使用し、CSV 形式はごく小さなディメンションテーブルにのみ使用します。
- プレフィックスベースのパーティショニングを使用して、パーティションプルーニングを活用します。つまり、データレイク内のパーティションにキー指定されたフィルターを使用します。
- Redshift Spectrum は大規模なリクエストを処理するために自動的にスケールされるため、Redshift Spectrum ではできるだけ多くの処理を行います (述語のプッシュダウンなど)。
- 頻繁にフィルタリングされる列のパーティションファイルには注意してください。データが 1 つ以上のフィルタリングされた列によってパーティション分割されている場合、Redshift Spectrum はパーティションプルーニングを活用し、不要なパーティションやファイルのスキャンをスキップできます。一般的な方法では、時間に基づいてデータをパーティション化します。
- 次のクエリを使用して、パーティションの有効性と Redshift Spectrum クエリの効率性を確認できます。

```
Select query,
           segment,
           max(assigned_partitions) as total_partitions,
           max(qualified_partitions) as qualified_partitions
From svl_s3partition
Where query=pg_last_query_id()
Group by 1,2;
```

前述のクエリは以下を示しています。

- total_partitions – によって認識されるパーティションの数 AWS Glue Data Catalog
- qualified_partitions – Redshift Spectrum クエリでアクセスされる Amazon Simple Storage Service (Amazon S3) のプレフィックスの数

- また、SVL_S3QUERY_SUMMARY システムテーブルをチェックして、パーティションの有効性と Redshift Spectrum クエリの効率性を確認することもできます。これを行うには、以下のステートメントを実行します。

```
Select *
From svl_s3query_summary
Where query=pg_last_query_id();
```

  前述のクエリは、パーティションプルーニングの効率性を示すファイルに加えて、is_partitioned、s3_scanned_rows/bytes、および s3_returned_rows/bytes の値を含むさらに多くの情報を返します。

Redshift Spectrum での述語プッシュダウン

述語プッシュダウンを使用すると、Amazon Redshift クラスターでのリソースの消費を回避できます。多くの SQL オペレーションを Redshift Spectrum レイヤーにプッシュダウンできます。これを可能な限り活用することをお勧めします。

以下に留意してください。

- Redshift Spectrum レイヤー内では、次のようないくつかのタイプの SQL オペレーションを完全に評価できます。
  - GROUP BY 句
  - 比較条件とパターンマッチング条件 (例: LIKE)
  - 集計関数 (例: COUNT、SUM、AVG、MIN、MAX)
  - regex_replace、to_upper、date_trunc、およびその他の関数
- DISTINCT や ORDER BY を含む一部のオペレーションを Redshift Spectrum レイヤーにプッシュすることはできません。ソートはリーダーノードで実行されるため、可能であればクエリの最上位レベルでのみ ORDER BY を実行してください。
- EXPLAIN クエリプランを調べて、述語プッシュダウンが有効かどうかを確認します。EXPLAIN コマンドで Redshift Spectrum 部分を検索するには、以下のステップを参照してください。
  - S3 Seq Scan
  - S3 HashAggregate
  - S3 Query Scan
  - Seq Scan PartitionInfo

  - Partition Loop
- クエリで使用する列数を最小限にします。Redshift Spectrum では、データが Parquet 形式または ORC 形式の場合、スキャンする列を減らすことができます。
- 並列処理とパーティション排除のためにパーティションをフル活用し、可能であればファイルサイズを 64 MB 以上に維持します。
- `CREATE EXTERNAL TABLE` または `ALTER TABLE` を使用する場合は、`TABLE PROPERTIES 'numRows'='nnn'` を設定します。Amazon Redshift は、外部テーブルを分析して、クエリオプティマイザがクエリプランを生成するために使用するテーブル統計を生成することはありません。統計情報が設定されていない場合、Amazon Redshift は、外部テーブルの方が大きくローカルテーブルの方が小さいということを前提にします。

Redshift Spectrum のクエリ調整のヒント

クエリを調整するときは、次の点に注意することをお勧めします。

- Amazon Redshift クラスターがクエリにエンゲージできる Redshift Spectrum ノードの数は、クラスター内のスライスの数に関連付けられます。
- クラスターのサイズを変更すると、クラスターのローカルコンピューティングプロファイル、ストレージプロファイル、Amazon S3 データレイククエリのクエリ機能にメリットがあります。
- Amazon Redshift クエリプランナーは、述語と集計を可能な限り Redshift Spectrum クエリレイヤーにプッシュします。
- Amazon S3 から大量のデータが返されると、クラスターのリソースによって処理が制限されます。
- Redshift Spectrum は大規模なリクエストを処理するように自動的にスケールするため、Redshift Spectrum レイヤーに処理をプッシュできるたびに全体的なパフォーマンスが向上します。

リソース

- Amazon Redshift のベストプラクティス (Amazon Redshift ドキュメント)
- Amazon Redshift クエリの設計のベストプラクティス (Amazon Redshift ドキュメント)
- クエリパフォーマンスのチューニング (Amazon Redshift ドキュメント)
- クエリプラン (Amazon Redshift ドキュメント)
- Amazon Redshift Spectrum クエリパフォーマンスの向上 (Amazon Redshift ドキュメント)
- Amazon Redshift のクエリライフサイクルについて (AWS 規範ガイダンス)

ドキュメント履歴

以下の表は、本ガイドの重要な変更点について説明したものです。今後の更新に関する通知を受け取る場合は、RSS フィード をサブスクライブできます。

変更	説明	日付
AQUA を削除	Advanced Query Accelerator (AQUA) に関する情報を削除しました。	2024 年 6 月 14 日
初版発行	—	2023 年 2 月 3 日

AWS 規範ガイダンスの用語集

以下は、 AWS 規範ガイダンスが提供する戦略、ガイド、パターンで一般的に使用される用語です。エントリを提案するには、用語集の最後のフィードバックの提供リンクを使用します。

数字

7 Rs

アプリケーションをクラウドに移行するための 7 つの一般的な移行戦略。これらの戦略は、ガートナーが 2011 年に特定した 5 Rs に基づいて構築され、以下で構成されています。

- リファクタリング/アーキテクチャの再設計 — クラウドネイティブ特徴を最大限に活用して、俊敏性、パフォーマンス、スケーラビリティを向上させ、アプリケーションを移動させ、アーキテクチャを変更します。これには、通常、オペレーティングシステムとデータベースの移植が含まれます。例: オンプレミスの Oracle データベースを Amazon Aurora PostgreSQL 互換エディションに移行する。
- リプラットフォーム (リフトアンドリシェイプ) — アプリケーションをクラウドに移行し、クラウド機能を活用するための最適化レベルを導入します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの Oracle 用の Amazon Relational Database Service (Amazon RDS) に移行する。
- 再購入 (ドロップアンドショップ) — 通常、従来のライセンスから SaaS モデルに移行して、別の製品に切り替えます。例: 顧客関係管理 (CRM) システムを Salesforce.com に移行する。
- リホスト (リフトアンドシフト) — クラウド機能を活用するための変更を加えずに、アプリケーションをクラウドに移行します。例: お客様のオンプレミスの Oracle データベースを AWS クラウドの EC2 インスタンス上の Oracle に移行する。
- 再配置 (ハイパーバイザーレベルのリフトアンドシフト) – 新しいハードウェアを購入したり、アプリケーションを書き換えたり、既存の運用を変更したりすることなく、インフラストラクチャをクラウドに移行できます。オンプレミスプラットフォームから同じプラットフォームのクラウドサービスにサーバーを移行します。例： Microsoft Hyper-Vアプリケーションを に移行します AWS。
- 保持 (再アクセス) — アプリケーションをお客様のソース環境で保持します。これには、主要なリファクタリングを必要とするアプリケーションや、お客様がその作業を後日まで延期したいアプリケーション、およびそれらを移行するためのビジネス上の正当性がないため、お客様が保持するレガシーアプリケーションなどがあります。
- 廃止 — お客様のソース環境で不要になったアプリケーションを停止または削除します。

A

ABAC

「属性ベースのアクセス制御」をご覧ください。

抽象化されたサービス

「マネージドユーザー」をご覧ください。

ACID

「原子性、一貫性、分離性、耐久性 (ACID)」をご覧ください。

アクティブ/アクティブ移行

(双方向レプリケーションツールまたは二重書き込み操作を使用して) ソースデータベースとターゲットデータベースを同期させ、移行中に両方のデータベースが接続アプリケーションからのトランザクションを処理するデータベース移行方法。この方法では、1 回限りのカットオーバーの必要がなく、管理された小規模なバッチで移行できます。アクティブ/パッシブ移行よりも柔軟な方法ですが、さらに多くの作業が必要となります。

アクティブ/パッシブ移行

ソースデータベースとターゲットデータベースを同期させながら、データがターゲットデータベースにレプリケートされている間、接続しているアプリケーションからのトランザクションをソースデータベースのみで処理するデータベース移行方法。移行中、ターゲットデータベースはトランザクションを受け付けません。

集計関数

複数行に処理を行い、グループ全体を対象に単一の戻り値を計算する SQL 関数。集計関数の例としては、SUM や MAX などがあります。

AI

「人工知能」をご覧ください。

AIOps

「AI オペレーション」をご覧ください。

匿名化

データセット内の個人情報を完全に削除するプロセス。匿名化は個人のプライバシー保護に役立ちます。匿名化されたデータは、もはや個人データとは見なされません。

アンチパターン

繰り返し起こる問題に対して頻繁に用いられる解決策で、その解決策が逆効果であったり、効果がなかったり、代替案よりも効果が低かったりするもの。

アプリケーション制御

マルウェアからシステムを保護するために、承認されたアプリケーションのみを使用できるようにするセキュリティアプローチ。

アプリケーションポートフォリオ

アプリケーションの構築と維持にかかるコスト、およびそのビジネス価値を含む、組織が使用する各アプリケーションに関する詳細情報の集まり。この情報は、ポートフォリオの検出と分析プロセス の重要な要素であり、移行、モダナイズ、最適化するアプリケーションを特定し、優先順位を付けるのに役立ちます。

人工知能 (AI)

コンピューティングテクノロジーを使用し、学習、問題の解決、パターンの認識など、通常は人間に関連づけられる認知機能の実行に特化したコンピュータサイエンスの分野。詳細については、「人工知能 (AI) とは何ですか?」をご覧ください。

AI オペレーション (AIOps)

機械学習技術を使用して運用上の問題を解決し、運用上のインシデントと人の介入を減らし、サービス品質を向上させるプロセス。 AWS 移行戦略での AIOps の使用方法については、オペレーション統合ガイド を参照してください。

非対称暗号化

暗号化用のパブリックキーと復号用のプライベートキーから成る 1 組のキーを使用した、暗号化のアルゴリズム。パブリックキーは復号には使用されないため共有しても問題ありませんが、プライベートキーの利用は厳しく制限する必要があります。

原子性、一貫性、分離性、耐久性 (ACID)

エラー、停電、その他の問題が発生した場合でも、データベースのデータ有効性と運用上の信頼性を保証する一連のソフトウェアプロパティ。

属性ベースのアクセス制御 (ABAC)

部署、役職、チーム名など、ユーザーの属性に基づいてアクセス許可をきめ細かく設定する方法。詳細については、 AWS Identity and Access Management (IAM) ドキュメントの「 の ABAC AWS」を参照してください。

信頼できるデータソース

最も信頼性のある情報源とされるデータのプライマリーバージョンを保存する場所。匿名化、編集、仮名化など、データを処理または変更する目的で、信頼できるデータソースから他の場所にデータをコピーすることができます。

アベイラビリティーゾーン (AZ)

他のアベイラビリティーゾーンの障害から AWS リージョン 隔離され、同じリージョン内の他のアベイラビリティーゾーンへの低コストで低レイテンシーのネットワーク接続を提供する 内の別の場所。

AWS クラウド導入フレームワーク (AWS CAF)

組織がクラウドへの移行を成功させるための効率的で効果的な計画を立て AWS るための、 のガイドラインとベストプラクティスのフレームワークです。 AWS CAF は、ビジネス、人材、ガバナンス、プラットフォーム、セキュリティ、運用という 6 つの重点分野にガイダンスをまとめています。ビジネス、人材、ガバナンスの観点では、ビジネススキルとプロセスに重点を置き、プラットフォーム、セキュリティ、オペレーションの視点は技術的なスキルとプロセスに焦点を当てています。例えば、人材の観点では、人事 (HR)、人材派遣機能、および人材管理を扱うステークホルダーを対象としています。この観点から、 AWS CAF は、クラウド導入を成功させるための組織の準備に役立つ人材開発、トレーニング、コミュニケーションに関するガイダンスを提供します。詳細については、AWS CAF ウェブサイトと AWS CAF のホワイトペーパー を参照してください。

AWS ワークロード認定フレームワーク (AWS WQF)

データベース移行ワークロードを評価し、移行戦略を推奨し、作業見積もりを提供するツール。AWS WQF は AWS Schema Conversion Tool (AWS SCT) に含まれています。データベーススキーマとコードオブジェクト、アプリケーションコード、依存関係、およびパフォーマンス特性を分析し、評価レポートを提供します。

B

不正なボット

個人や組織に混乱や損害を与えることを目的としたボット。

BCP

「ビジネス継続性計画 (BCP)」をご覧ください。

動作グラフ

リソースの動作とインタラクションを経時的に示した、一元的なインタラクティブビュー。Amazon Detective の動作グラフを使用すると、失敗したログオンの試行、不審な API 呼び出し、その他同様のアクションを調べることができます。詳細については、Detective ドキュメントの「動作グラフのデータ」を参照してください。

ビッグエンディアンシステム

最上位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

二項分類

バイナリ結果 (2 つの可能なクラスのうちの 1 つ) を予測するプロセス。例えば、お客様の機械学習モデルで「この E メールはスパムですか、それともスパムではありませんか」などの問題を予測する必要があるかもしれません。または「この製品は書籍ですか、車ですか」などの問題を予測する必要があるかもしれません。

ブルームフィルター

要素がセットのメンバーであるかどうかをテストするために使用される、確率的でメモリ効率の高いデータ構造。

ブルー/グリーンデプロイ

それぞれが独立しているが、同一の環境を 2 つ作成するデプロイ戦略。現在のアプリケーションバージョンを 1 つの環境 (ブルー) で実行し、新しいアプリケーションバージョンを別の環境 (グリーン) で実行します。この戦略は、最小限の影響で迅速にロールバックするのに役立ちます。

ボット

インターネット経由で自動タスクを実行し、人間のアクティビティややり取りをシミュレートするソフトウェアアプリケーション。インターネット上の情報のインデックスを作成するウェブクローラーなど、一部のボットは有用または有益です。悪質なボットと呼ばれる他のボットの中には、個人や組織を混乱させたり、損害を与えたりすることを意図したものもあります。

ボットネット

マルウェアに感染しており、ボットハーダーまたはボットオペレーターと呼ばれる単一の当事者によって制御されているボットのネットワーク。ボットネットは、ボットとその影響力を拡大する仕組みとして、非常によく知られています。

ブランチ

コードリポジトリに含まれる領域。リポジトリに最初に作成するブランチは、メインブランチといいます。既存のブランチから新しいブランチを作成し、その新しいブランチで機能を開発した

り、バグを修正したりできます。機能を構築するために作成するブランチは、通常、機能ブランチと呼ばれます。機能をリリースする準備ができたら、機能ブランチをメインブランチに統合します。詳細については、「ブランチの概要」(GitHub ドキュメント) を参照してください。

ブレークグラスアクセス

例外的な状況では、承認されたプロセスを通じて、ユーザーが AWS アカウント 通常アクセス許可を持たない にすばやくアクセスできるようにします。詳細については、 AWS Well-Architected ガイダンスの「ブレークグラス手順の実装」インジケータを参照してください。

ブラウンフィールド戦略

環境の既存インフラストラクチャ。システムアーキテクチャにブラウンフィールド戦略を導入する場合、現在のシステムとインフラストラクチャの制約に基づいてアーキテクチャを設計します。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

バッファキャッシュ

アクセス頻度が最も高いデータが保存されるメモリ領域。

ビジネス能力

価値を生み出すためにビジネスが行うこと (営業、カスタマーサービス、マーケティングなど)。マイクロサービスのアーキテクチャと開発の決定は、ビジネス能力によって推進できます。詳細については、AWSでのコンテナ化されたマイクロサービスの実行ホワイトペーパーの「ビジネス機能を中心に組織化」セクションを参照してください。

ビジネス継続性計画 (BCP)

大規模移行など、中断を伴うイベントが運用に与える潜在的な影響に対処し、ビジネスを迅速に再開できるようにする計画。

C

CAF

「AWS クラウド導入フレームワーク」を参照してください

カナリアデプロイ

エンドユーザーへのバージョンリリースを、時間をかけて段階的に行うこと。確信が持てたら新規バージョンをデプロイして、現在のバージョン全体を置き換えます。

CCoE

「Cloud Center of Excellence」を参照してください。

CDC

「変更データキャプチャ」を参照してください。

変更データキャプチャ (CDC)

データソース (データベーステーブルなど) の変更を追跡し、その変更に関するメタデータを記録するプロセス。CDC は、ターゲットシステムでの変更を監査またはレプリケートして同期を維持するなど、さまざまな目的に使用できます。

カオスエンジニアリング

障害や破壊的なイベントを意図的に導入して、システムの耐障害性をテストすること。AWS Fault Injection Service (AWS FIS) を使用して、 AWS ワークロードにストレスを与え、その応答を評価する実験を実行できます。

CI/CD

「継続的インテグレーションと継続的デリバリー」を参照してください。

分類

予測を生成するのに役立つ分類プロセス。分類問題の機械学習モデルは、離散値を予測します。離散値は、常に互いに区別されます。例えば、モデルがイメージ内に車があるかどうかを評価する必要がある場合があります。

クライアント側の暗号化

ターゲットがデータ AWS のサービス を受信する前に、ローカルでデータを暗号化します。

Cloud Center of Excellence (CCoE)

クラウドのベストプラクティスの作成、リソースの移動、移行のタイムラインの確立、大規模変革を通じて組織をリードするなど、組織全体のクラウド導入の取り組みを推進する学際的なチーム。詳細については、 AWS クラウド エンタープライズ戦略ブログの CCoE 投稿を参照してください。

クラウドコンピューティング

リモートデータストレージと IoT デバイス管理に通常使用されるクラウドテクノロジー。クラウドコンピューティングは、一般的に、エッジコンピューティングに接続されています。

クラウド運用モデル

IT 組織において、1 つ以上のクラウド環境を構築、成熟、最適化するために使用される運用モデル。詳細については、「クラウド運用モデルの構築」を参照してください。

導入のクラウドステージ

組織が、 AWS クラウドへの移行時に通常実行する 4 つの段階。

- プロジェクト — 概念実証と学習を目的として、クラウド関連のプロジェクトをいくつか実行する
- 基礎固め — お客様のクラウドの導入を拡大するための基礎的な投資 (ランディングゾーンの作成、CCoE の定義、運用モデルの確立など)
- 移行 — 個々のアプリケーションの移行
- 再発明 — 製品とサービスの最適化、クラウドでのイノベーション

これらのステージは、 AWS クラウド エンタープライズ戦略ブログのブログ記事「クラウドファーストへのジャーニー」と「導入のステージ」で Stephen Orban によって定義されました。移行戦略との関連性については、 AWS 「移行準備ガイド」を参照してください。

CMDB

「構成管理データベース (CMDB)」を参照してください。

コードリポジトリ

ソースコードやその他の資産 (ドキュメント、サンプル、スクリプトなど) が保存され、バージョン管理プロセスを通じて更新される場所。一般的なクラウドリポジトリには、GitHub や Bitbucket Cloud があります。コードの各バージョンはブランチと呼ばれます。マイクロサービスの構造では、各リポジトリは 1 つの機能専用です。1 つの CI/CD パイプラインで複数のリポジトリを使用できます。

コールドキャッシュ

空である、または、かなり空きがある、もしくは、古いデータや無関係なデータが含まれているバッファキャッシュ。データベースインスタンスはメインメモリまたはディスクから読み取る必要があり、バッファキャッシュから読み取るよりも時間がかかるため、パフォーマンスに影響します。

コールドデータ

めったにアクセスされず、通常は過去のデータです。この種類のデータをクエリする場合、通常は低速なクエリでも問題ありません。このデータを低パフォーマンスで安価なストレージ階層またはクラスに移動すると、コストを削減することができます。

コンピュータビジョン (CV)

機械学習を使用してデジタルイメージやビデオといった、ビジュアル形式の情報を分析および抽出する AI の分野。例えば、Amazon SageMaker AI では、CV 用の画像処理アルゴリズムを利用できます。

設定ドリフト

ワークロードにおいて、設定が想定した状態から変化すること。これによって、ワークロードが非準拠になる可能性がありますが、この状態は、徐々に生じ、意図的なものではありません。

構成管理データベース (CMDB)

データベースとその IT 環境 (ハードウェアとソフトウェアの両方のコンポーネントとその設定を含む) に関する情報を保存、管理するリポジトリ。通常、CMDB のデータは、移行のポートフォリオの検出と分析の段階で使用します。

コンフォーマンスパック

コンプライアンスチェックとセキュリティチェックをカスタマイズするためにアセンブルできる AWS Config ルールと修復アクションのコレクション。YAML テンプレートを使用して、コンフォーマンスパックを AWS アカウント および リージョンの単一のエンティティとしてデプロイすることも、組織全体にデプロイすることもできます。詳細については、 AWS Config ドキュメントの「コンフォーマンスパック」を参照してください。

継続的インテグレーションと継続的デリバリー (CI/CD)

ソフトウェアリリースプロセスのソース、ビルド、テスト、ステージング、本番の各ステージを自動化するプロセス。CI/CD は一般的にパイプラインと呼ばれます。プロセスの自動化、生産性の向上、コード品質の向上、配信の加速化を可能にします。詳細については、「継続的デリバリーの利点」を参照してください。CD は継続的デプロイ (Continuous Deployment) の略語でもあります。詳細については「継続的デリバリーと継続的なデプロイ」を参照してください。

CV

「コンピュータビジョン」を参照してください。

D

保管中のデータ

ストレージ内にあるデータなど、常に自社のネットワーク内にあるデータ。

データ分類

ネットワーク内のデータを重要度と機密性に基づいて識別、分類するプロセス。データに適した保護および保持のコントロールを判断する際に役立つため、あらゆるサイバーセキュリティのリスク管理戦略において重要な要素です。データ分類は、 AWS Well-Architected フレームワークのセキュリティの柱のコンポーネントです。詳細については、「データ分類」を参照してください。

データドリフト

実稼働データと ML モデルのトレーニングに使用されたデータとの間に有意な差異が生じたり、入力データが時間の経過と共に有意に変化したりすることです。データドリフトは、ML モデル予測の全体的な品質、精度、公平性を低下させる可能性があります。

転送中のデータ

ネットワーク内 (ネットワークリソース間など) を活発に移動するデータ。

データメッシュ

非一元的で分散型のデータ所有権を持つとともに、一元的な管理およびガバナンスを行えるアーキテクチャフレームワーク。

データ最小化

厳密に必要なデータのみを収集し、処理するという原則。でデータ最小化を実践 AWS クラウドすることで、プライバシーリスク、コスト、分析のカーボンフットプリントを削減できます。

データ境界

AWS 環境内の一連の予防ガードレール。信頼された ID のみが、期待されるネットワークから信頼されたリソースにアクセスできるようにします。詳細については、「 でのデータ境界の構築 AWS」を参照してください。

データの前処理

raw データをお客様の機械学習モデルで簡単に解析できる形式に変換すること。データの前処理とは、特定の列または行を削除して、欠落している、矛盾している、または重複する値に対処することを意味します。

データ出所

データの生成、送信、保存の方法など、データのライフサイクル全体を通じてデータの出所と履歴を追跡するプロセス。

データ件名

データを収集、処理している個人。

データウェアハウス

分析などのビジネスインテリジェンスをサポートするデータ管理システム。データウェアハウスには、一般的に、大量の履歴データが含まれており、多くの場合、それらはクエリや分析に使用されます。

データベース定義言語 (DDL)

データベース内のテーブルやオブジェクトの構造を作成または変更するためのステートメントまたはコマンド。

データベース操作言語 (DML)

データベース内の情報を変更 (挿入、更新、削除) するためのステートメントまたはコマンド。

DDL

「データベース定義言語」を参照してください。

ディープアンサンブル

予測のために複数の深層学習モデルを組み合わせます。ディープアンサンブルを使用して、より正確な予測を取得したり、予測の不確実性を推定したりできます。

深層学習

人工ニューラルネットワークの複数層を使用して、入力データと対象のターゲット変数の間のマッピングを識別する機械学習サブフィールド。

多層防御

一連のセキュリティメカニズムとコントロールをコンピュータネットワーク全体に層状に重ねて、ネットワークとその内部にあるデータの機密性、整合性、可用性を保護する情報セキュリティの手法。この戦略を に採用するときは AWS、リソースの保護に役立つように、 AWS Organizations 構造の異なるレイヤーに複数のコントロールを追加します。たとえば、多層防御アプローチでは、多要素認証、ネットワークセグメンテーション、暗号化を組み合わせることができます。

委任管理者

では AWS Organizations、互換性のあるサービスが AWS メンバーアカウントを登録して組織のアカウントを管理し、そのサービスのアクセス許可を管理できます。このアカウントを、そのサービスの委任管理者と呼びます。詳細、および互換性のあるサービスの一覧は、 AWS

Organizations ドキュメントの「AWS Organizationsで使用できるサービス」を参照してください。

トラブルシューティング

アプリケーション、新機能、コードの修正をターゲットの環境で利用できるようにするプロセス。デプロイでは、コードベースに変更を施した後、アプリケーションの環境でそのコードベースを構築して実行します。

開発環境

「環境」を参照してください。

検出管理

イベントが発生したときに、検出、ログ記録、警告を行うように設計されたセキュリティコントロール。これらのコントロールは副次的な防衛手段であり、実行中の予防的コントロールをすり抜けたセキュリティイベントをユーザーに警告します。詳細については、「 AWSでのセキュリティコントロールの実装」の「検出的コントロール」を参照してください。

開発バリューストリームマッピング (DVSM)

ソフトウェア開発ライフサイクルのスピードと品質に悪影響を及ぼす制約を特定し、優先順位を付けるために使用されるプロセス。DVSM は、もともとリーンマニュファクチャリング・プラクティスのために設計されたバリューストリームマッピング・プロセスを拡張したものです。ソフトウェア開発プロセスを通じて価値を創造し、動かすために必要なステップとチームに焦点を当てています。

デジタルツイン

建物、工場、産業機器、生産ラインなど、現実世界のシステムを仮想的に表現したものです。デジタルツインは、予知保全、リモートモニタリング、生産最適化をサポートします。

ディメンションテーブル

スタースキーマにおいて、ファクトテーブルの定量データに関するデータ属性が含まれる小さいテーブル。ディメンションテーブルの属性は、通常、テキストフィールド、またはテキストのように扱える個別の数値で示されます。これらの属性は、一般的に、クエリの制約、フィルタリング、結果セットのラベル付けに使用されます。

ディザスタ

ワークロードまたはシステムが、導入されている主要な場所でのビジネス目標の達成を妨げるイベント。これらのイベントは、自然災害、技術的障害、または意図しない設定ミスやマルウェア攻撃などの人間の行動の結果である場合があります。

ディザスタリカバリ (DR)

ディザスタによるダウンタイムとデータ損失を最小限に抑えるための戦略とプロセス。詳細については、 AWS Well-Architected フレームワークの「 でのワークロードのディザスタリカバリ AWS: クラウドでのリカバリ」を参照してください。

DML

「データベース操作言語」を参照してください。

ドメイン駆動型設計

各コンポーネントが提供している変化を続けるドメイン、またはコアビジネス目標にコンポーネントを接続して、複雑なソフトウェアシステムを開発するアプローチ。この概念は、エリック・エヴァンスの著書、Domain-Driven Design: Tackling Complexity in the Heart of Software (ドメイン駆動設計:ソフトウェアの中心における複雑さへの取り組み) で紹介されています (ボストン: Addison-Wesley Professional、2003)。strangler fig パターンでドメイン駆動型設計を使用する方法の詳細については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

DR

「ディザスタリカバリ」を参照してください。

ドリフト検出

ベースライン設定からの偏差を追跡します。たとえば、 AWS CloudFormation を使用してシステムリソースのドリフトを検出したり、 を使用して AWS Control Tower 、ガバナンス要件への準拠に影響するランディングゾーンの変更を検出したりできます。

DVSM

「開発バリューストリームマッピング」を参照してください。

E

EDA

「探索的データ分析」を参照してください。

EDI

「電子データ交換」を参照してください。

エッジコンピューティング

IoT ネットワークのエッジにあるスマートデバイスの計算能力を高めるテクノロジー。クラウドコンピューティングと比較すると、エッジコンピューティングは通信レイテンシーを短縮し、応答時間を改善できます。

電子データ交換 (EDI)

組織間で行う、ビジネスドキュメントの自動交換。詳細については、「電子データ交換とは」を参照してください。

暗号化

人間が読み取り可能なプレーンテキストデータを暗号文に変換するコンピューティング処理。

暗号化キー

暗号化アルゴリズムが生成した、ランダム化されたビットからなる暗号文字列。キーの長さは決まっておらず、各キーは予測できないように、一意になるように設計されています。

エンディアン

コンピュータメモリにバイトが格納される順序。ビッグエンディアンシステムでは、最上位バイトが最初に格納されます。リトルエンディアンシステムでは、最下位バイトが最初に格納されます。

エンドポイント

「サービスエンドポイント」を参照してください。

エンドポイントサービス

仮想プライベートクラウド (VPC) 内でホストして、他のユーザーと共有できるサービス。を使用してエンドポイントサービスを作成し AWS PrivateLink 、他の AWS アカウント または AWS Identity and Access Management (IAM) プリンシパルにアクセス許可を付与できます。これらのアカウントまたはプリンシパルは、インターフェイス VPC エンドポイントを作成することで、エンドポイントサービスにプライベートに接続できます。詳細については、Amazon Virtual Private Cloud (Amazon VPC) ドキュメントの「エンドポイントサービスを作成する」を参照してください。

エンタープライズリソースプランニング (ERP)

エンタープライズの主要なビジネスプロセス (会計、MES、プロジェクト管理など) を自動化および管理するシステム。

エンベロープ暗号化

暗号化キーを、別の暗号化キーを使用して暗号化するプロセス。詳細については、 AWS Key Management Service (AWS KMS) ドキュメントの「エンベロープ暗号化」を参照してください。

環境

実行中のアプリケーションのインスタンス。クラウドコンピューティングにおける一般的な環境の種類は以下のとおりです。

- 開発環境 — アプリケーションのメンテナンスを担当するコアチームのみが使用できる、実行中のアプリケーションのインスタンス。開発環境は、上位の環境に昇格させる変更をテストするときに使用します。このタイプの環境は、テスト環境と呼ばれることもあります。
- 下位環境 — 初期ビルドやテストに使用される環境など、アプリケーションのすべての開発環境。
- 本番環境 — エンドユーザーがアクセスできる、実行中のアプリケーションのインスタンス。CI/CD パイプラインでは、本番環境が最後のデプロイ環境になります。
- 上位環境 — コア開発チーム以外のユーザーがアクセスできるすべての環境。これには、本番環境、本番前環境、ユーザー承認テスト環境などが含まれます。

エピック

アジャイル方法論で、お客様の作業の整理と優先順位付けに役立つ機能カテゴリ。エピックでは、要件と実装タスクの概要についてハイレベルな説明を提供します。例えば、 AWS CAF セキュリティエピックには、ID とアクセスの管理、検出コントロール、インフラストラクチャセキュリティ、データ保護、インシデント対応が含まれます。 AWS 移行戦略のエピックの詳細については、プログラム実装ガイドを参照してください。

ERP

「エンタープライズリソース計画」を参照してください。

探索的データ分析 (EDA)

データセットを分析してその主な特性を理解するプロセス。お客様は、データを収集または集計してから、パターンの検出、異常の検出、および前提条件のチェックのための初期調査を実行します。EDA は、統計の概要を計算し、データの可視化を作成することによって実行されます。

F

ファクトテーブル

スタースキーマの中央にあるテーブル。ビジネスオペレーションに関する定量的データが保存されます。一般的に、ファクトテーブルは、2 種類の列で構成されます。1 つは測定値が含まれる列、もう 1 つはディメンションテーブルへの外部キーが含まれる列です。

フェイルファスト

開発ライフサイクルを短縮するために、頻繁かつ段階的にテストを行う哲学であり、アジャイルアプローチでは、この考え方がきわめて重要です。

障害分離境界

では AWS クラウド、障害の影響を制限し、ワークロードの耐障害性を高めるのに役立つアベイラビリティーゾーン AWS リージョン、コントロールプレーン、データプレーンなどの境界。詳細については、「AWS 障害分離境界」を参照してください。

機能ブランチ

「ブランチ」を参照してください。

特徴量

お客様が予測に使用する入力データ。例えば、製造コンテキストでは、特徴量は製造ラインから定期的にキャプチャされるイメージの可能性もあります。

特徴量重要度

モデルの予測に対する特徴量の重要性。これは通常、Shapley Additive Deskonations (SHAP) や積分勾配など、さまざまな手法で計算できる数値スコアで表されます。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

機能変換

追加のソースによるデータのエンリッチ化、値のスケーリング、単一のデータフィールドからの複数の情報セットの抽出など、機械学習プロセスのデータを最適化すること。これにより、機械学習モデルはデータの恩恵を受けることができます。例えば、「2021-05-27 00:15:37」の日付を「2021 年」、「5 月」、「木」、「15」に分解すると、学習アルゴリズムがさまざまなデータコンポーネントに関連する微妙に異なるパターンを学習するのに役立ちます。

数ショットプロンプト

LLM に、タスクと望ましい出力を示す例を少数提示した後に、類似のタスクを実行させること。この手法は、プロンプトに記述された例 (ショット) からモデルが学習する「インコンテキスト学

習」の一種です。数ショットプロンプトは、特定のフォーマット、推論、専門知識が必要なタスクに効果的です。「ゼロショットプロンプト」も参照してください。

FGAC

「きめ細かなアクセス制御」を参照してください。

きめ細かなアクセス制御 (FGAC)

複数の条件を使用してアクセス要求を許可または拒否すること。

フラッシュカット移行

変更データのキャプチャによる継続的なデータ複製を利用して、段階的なアプローチではなく、可能な限り短時間でデータを移行するデータベース移行方法。目的はダウンタイムを最小限に抑えることです。

FM

「基盤モデル」を参照してください。

基盤モデル (FM)

大規模な深層学習ニューラルネットワークであり、一般化およびラベル付けされていないデータからなる大規模データセットでトレーニングされています。FM により、言語理解、テキストおよび画像生成、自然言語での会話といった、一般的な各種タスクを実行できます。詳細については、「基盤モデルとは何ですか?」を参照してください。

G

生成 AI

AI モデルのサブセット。大量のデータでトレーニングされており、シンプルなテキストプロンプトを使用して、画像、動画、テキスト、オーディオなどの新しいコンテンツやアーティファクトを作成できます。詳細については、「生成 AI とは何ですか?」を参照してください。

ジオブロッキング

「地理的制限」を参照してください。

地理的制限 (ジオブロッキング)

特定の国のユーザーがコンテンツ配信にアクセスできないようにするための、Amazon CloudFront のオプション。アクセスを許可する国と禁止する国は、許可リストまたは禁止リスト

を使って指定します。詳細については、CloudFront ドキュメントの「コンテンツの地理的ディストリビューションの制限」を参照してください。

Gitflow ワークフロー

下位環境と上位環境が、ソースコードリポジトリでそれぞれ異なるブランチを使用する方法。Gitflow ワークフローは古いと見なされている方法であり、トランクベースのワークフローは推奨されている新しい方法です。

ゴールデンイメージ

システムまたはソフトウェアのスナップショットであり、システムまたはソフトウェアの新規インスタンスをデプロイするテンプレートとして使用されます。製造の例で言えば、ゴールデンイメージを使用すると、複数のデバイスにソフトウェアをプロビジョニングして、デバイス製造オペレーションの速度、スケーラビリティ、生産性を向上させることができます。

グリーンフィールド戦略

新しい環境に既存のインフラストラクチャが存在しないこと。システムアーキテクチャにグリーンフィールド戦略を導入する場合、既存のインフラストラクチャ (別名ブラウンフィールド) との互換性の制約を受けることなく、あらゆる新しいテクノロジーを選択できます。既存のインフラストラクチャを拡張している場合は、ブラウンフィールド戦略とグリーンフィールド戦略を融合させることもできます。

ガードレール

組織単位 (OU) 全般のリソース、ポリシー、コンプライアンスを管理するのに役立つ概略的なルール。予防ガードレールは、コンプライアンス基準に一致するようにポリシーを実施します。これらは、サービスコントロールポリシーと IAM アクセス許可の境界を使用して実装されます。検出ガードレールは、ポリシー違反やコンプライアンス上の問題を検出し、修復のためのアラートを発信します。これらは AWS Config、、Amazon GuardDuty AWS Security Hub CSPM、、 AWS Trusted Advisor Amazon Inspector、およびカスタム AWS Lambda チェックを使用して実装されます。

H

HA

「高可用性」を参照してください。

異種混在データベースの移行

別のデータベースエンジンを使用するターゲットデータベースへお客様の出典データベースの移行 (例えば、Oracle から Amazon Aurora)。異種間移行は通常、アーキテクチャの再設計作業の一部であり、スキーマの変換は複雑なタスクになる可能性があります。AWS は、スキーマの変換に役立つ AWS SCTを提供します。

高可用性 (HA)

課題や災害が発生した場合に、介入なしにワークロードを継続的に運用できること。HA システムは、自動的にフェイルオーバーし、一貫して高品質のパフォーマンスを提供し、パフォーマンスへの影響を最小限に抑えながらさまざまな負荷や障害を処理するように設計されています。

ヒストリアンのモダナイゼーション

製造業のニーズによりよく応えるために、オペレーションテクノロジー (OT) システムをモダナイズし、アップグレードするためのアプローチ。ヒストリアンは、工場内のさまざまなソースからデータを収集して保存するために使用されるデータベースの一種です。

ホールドアウトデータ

機械学習モデルのトレーニング用データセットから保留される、ラベル付き履歴データの一部。ホールドアウトデータを使用すると、モデル予測をホールドアウトデータと比較して、モデルのパフォーマンスを評価できます。

同種データベースの移行

お客様の出典データベースを、同じデータベースエンジンを共有するターゲットデータベース (Microsoft SQL Server から Amazon RDS for SQL Server など) に移行する。同種間移行は、通常、リホストまたはリプラットフォーム化の作業の一部です。ネイティブデータベースユーティリティを使用して、スキーマを移行できます。

ホットデータ

リアルタイムデータや最近の翻訳データなど、頻繁にアクセスされるデータ。通常、このデータには高速なクエリ応答を提供する高性能なストレージ階層またはクラスが必要です。

ホットフィックス

本番環境の重大な問題を修正するために緊急で配布されるプログラム。緊急性が高いため、通常の DevOps のリリースワークフローからは外れた形で実施されます。

ハイパーケア期間

カットオーバー直後、移行したアプリケーションを移行チームがクラウドで管理、監視して問題に対処する期間。通常、この期間は 1～4 日です。ハイパーケア期間が終了すると、アプリケーションに対する責任は一般的に移行チームからクラウドオペレーションチームに移ります。

I

IaC

「Infrastructure as Code」を参照してください。

ID ベースのポリシー

AWS クラウド 環境内のアクセス許可を定義する 1 つ以上の IAM プリンシパルにアタッチされたポリシー。

アイドル状態のアプリケーション

90 日間の平均的な CPU およびメモリ使用率が 5～20% のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するか、オンプレミスに保持するのが一般的です。

IIoT

「インダストリアル IoT」を参照してください。

イミュータブルインフラストラクチャ

既存インフラストラクチャの更新、パッチ適用、変更などを行わずに、本番環境ワークロードに使用する新規インフラストラクチャをデプロイするモデル。本質的に、イミュータブルインフラストラクチャは、ミュータブルインフラストラクチャよりも一貫性、信頼性、予測性に優れています。詳細については、 AWS Well-Architected フレームワークにある「イミュータブルインフラストラクチャを使用してデプロイする」のベストプラクティスを参照してください。

インバウンド (受信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーションの外部からネットワーク接続を受け入れ、検査し、ルーティングする VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

増分移行

アプリケーションを 1 回ですべてカットオーバーするのではなく、小さい要素に分けて移行するカットオーバー戦略。例えば、最初は少数のマイクロサービスまたはユーザーのみを新しいシステムに移行する場合があります。すべてが正常に機能することを確認できたら、残りのマイクロサービスやユーザーを段階的に移行し、レガシーシステムを廃止できるようにします。この戦略により、大規模な移行に伴うリスクが軽減されます。

インダストリー 4.0

2016 年に Klaus Schwab 氏が提唱した用語で、接続、リアルタイムデータ、オートメーション、分析、AI/ML の進歩による、ビジネスプロセスのモダナイズを意味します。

インフラストラクチャ

アプリケーションの環境に含まれるすべてのリソースとアセット。

Infrastructure as Code (IaC)

アプリケーションのインフラストラクチャを一連の設定ファイルを使用してプロビジョニングし、管理するプロセス。IaC は、新しい環境を再現可能で信頼性が高く、一貫性のあるものにするため、インフラストラクチャを一元的に管理し、リソースを標準化し、スケールを迅速に行えるように設計されています。

インダストリアル IoT (IIoT)

製造、エネルギー、自動車、ヘルスケア、ライフサイエンス、農業などの産業部門におけるインターネットに接続されたセンサーやデバイスの使用。詳細については、「インダストリアル IoT (IIoT) デジタルトランスフォーメーション戦略の構築」を参照してください。

インスペクション VPC

AWS マルチアカウントアーキテクチャでは、VPC (同一または異なる 内 AWS リージョン)、インターネット、オンプレミスネットワーク間のネットワークトラフィックの検査を管理する一元化された VPCs。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

IoT

インターネットまたはローカル通信ネットワークを介して他のデバイスやシステムと通信する、センサーまたはプロセッサが組み込まれた接続済み物理オブジェクトのネットワーク。詳細については、「IoT とは」を参照してください。

解釈可能性

機械学習モデルの特性で、モデルの予測がその入力にどのように依存するかを人間が理解できる度合いを表します。詳細については、「 を使用した機械学習モデルの解釈可能性 AWS」を参照してください。

IoT

「IoT」を参照してください。

IT 情報ライブラリ (ITIL)

IT サービスを提供し、これらのサービスをビジネス要件に合わせるための一連のベストプラクティス。ITIL は ITSM の基盤を提供します。

IT サービス管理 (ITSM)

組織の IT サービスの設計、実装、管理、およびサポートに関連する活動。クラウドオペレーションと ITSM ツールの統合については、オペレーション統合ガイドを参照してください。

ITIL

「IT 情報ライブラリ」を参照してください。

ITSM

「IT サービス管理」を参照してください。

L

ラベルベースアクセス制御 (LBAC)

強制アクセス制御 (MAC) の実装で、ユーザーとデータ自体にそれぞれセキュリティラベル値が明示的に割り当てられます。ユーザーセキュリティラベルとデータセキュリティラベルが交差する部分によって、ユーザーに表示される行と列が決まります。

ランディングゾーン

ランディングゾーンは、スケーラブルで安全な、適切に設計されたマルチアカウント AWS 環境です。これは、組織がセキュリティおよびインフラストラクチャ環境に自信を持ってワークロードとアプリケーションを迅速に起動してデプロイできる出発点です。ランディングゾーンの詳細については、「安全でスケーラブルなマルチアカウント AWS 環境のセットアップ」を参照してください。

大規模言語モデル (LLM)

大量のデータで事前トレーニングされた深層学習 AI モデル。LLM では、質問への回答、ドキュメントの要約、他言語へのテキスト翻訳、文を完成させるなど、さまざまなタスクを実行できます。詳細については、「大規模言語モデル (LLM) とは何ですか?」を参照してください。

大規模な移行

300 台以上のサーバの移行。

LBAC

「ラベルベースアクセス制御」を参照してください。

最小特権

タスクの実行には必要最低限の権限を付与するという、セキュリティのベストプラクティス。詳細については、IAM ドキュメントの「最小特権アクセス許可を適用する」を参照してください。

リフトアンドシフト

「7 Rs」を参照してください。

リトルエンディアンシステム

最下位バイトを最初に格納するシステム。「エンディアン性」もご覧ください。

LLM

「大規模言語モデル」を参照してください。

下位環境

「環境」を参照してください。

M

機械学習 (ML)

パターン認識と学習にアルゴリズムと手法を使用する人工知能の一種。ML は、モノのインターネット (IoT) データなどの記録されたデータを分析して学習し、パターンに基づく統計モデルを生成します。詳細については、「機械学習」を参照してください。

メインブランチ

「ブランチ」を参照してください。

マルウェア

コンピュータのセキュリティやプライバシーを侵害するように設計されたソフトウェア。マルウェアは、コンピュータシステムの中断、機密情報の漏洩、不正アクセスを招く可能性があります。マルウェアの例には、ウイルス、ワーム、ランサムウェア、トロイの木馬、スパイウェア、キーロガーなどがあります。

マネージドサービス

AWS のサービス はインフラストラクチャレイヤー、オペレーティングシステム、プラットフォーム AWS を運用し、エンドポイントにアクセスしてデータを保存および取得します。マネージドサービスの例として、Amazon Simple Storage Service (Amazon S3) と Amazon DynamoDB が挙げられます。このサービスは、抽象化されたサービスとも呼ばれます。

製造実行システム (MES)

生産プロセスを追跡、モニタリング、文書化、制御するソフトウェアシステムであり、工場では、これによって、原材料から製品を完成させます。

MAP

「Migration Acceleration Program」を参照してください。

メカニズム

ツールを作成してその導入を推進し、導入結果を調べて調整を行うための包括的なプロセス。メカニズムとは、運用中にそれ自体を強化し改善するサイクルを意味します。詳細については、AWS 「 Well-Architected フレームワーク」の「メカニズムの構築」を参照してください。

メンバーアカウント

組織の一部である管理アカウント AWS アカウント 以外のすべて AWS Organizations。 アカウントが 組織のメンバーになることができるのは、一度に 1 つのみです。

MES

「製造実行システム」を参照してください。

Message Queuing Telemetry Transport (MQTT)

発行/サブスクライブのパターンに基づく、軽量のマシンツーマシン (M2M) 通信プロトコルであり、リソースに限りのある IoT デバイスに使用されます。

マイクロサービス

明確に定義された API を介して通信し、通常は小規模な自己完結型のチームが所有する、小規模で独立したサービスです。例えば、保険システムには、販売やマーケティングなどのビジネス

機能、または購買、請求、分析などのサブドメインにマッピングするマイクロサービスが含まれる場合があります。マイクロサービスの利点には、俊敏性、柔軟なスケーリング、容易なデプロイ、再利用可能なコード、回復力などがあります。詳細については、AWS「サーバーレスサービスを使用したマイクロサービスの統合」を参照してください。

マイクロサービスアーキテクチャ

各アプリケーションプロセスをマイクロサービスとして実行する独立したコンポーネントを使用してアプリケーションを構築するアプローチ。これらのマイクロサービスは、軽量 API を使用して、明確に定義されたインターフェイスを介して通信します。このアーキテクチャの各マイクロサービスは、アプリケーションの特定の機能に対する需要を満たすように更新、デプロイ、およびスケーリングできます。詳細については、「 でのマイクロサービスの実装 AWS」を参照してください。

Migration Acceleration Program (MAP)

組織がクラウドに移行するための強力な運用基盤を構築し、移行の初期コストを相殺するのに役立つコンサルティングサポート、トレーニング、サービスを提供する AWS プログラム。MAP には、組織的な方法でレガシー移行を実行するための移行方法論と、一般的な移行シナリオを自動化および高速化する一連のツールが含まれています。

大規模な移行

アプリケーションポートフォリオの大部分を次々にクラウドに移行し、各ウェーブでより多くのアプリケーションを高速に移動させるプロセス。この段階では、以前の段階から学んだベストプラクティスと教訓を使用して、移行ファクトリー チーム、ツール、プロセスのうち、オートメーションとアジャイルデリバリーによってワークロードの移行を合理化します。これは、AWS 移行戦略 の第 3 段階です。

移行ファクトリー

自動化された俊敏性のあるアプローチにより、ワークロードの移行を合理化する部門横断的なチーム。移行ファクトリーチームには、通常、運用、ビジネスアナリストおよび所有者、移行エンジニア、デベロッパー、およびスプリントで作業する DevOps プロフェッショナルが含まれます。エンタープライズアプリケーションポートフォリオの 20～50% は、ファクトリーのアプローチによって最適化できる反復パターンで構成されています。詳細については、このコンテンツセットの移行ファクトリーに関する解説とCloud Migration Factory ガイドを参照してください。

移行メタデータ

移行を完了するために必要なアプリケーションおよびサーバーに関する情報。移行パターンごとに、異なる一連の移行メタデータが必要です。移行メタデータの例としては、ターゲットサブネット、セキュリティグループ、 AWS アカウントなどがあります。

移行パターン

移行戦略、移行先、および使用する移行アプリケーションまたはサービスを詳述する、反復可能な移行タスク。例: AWS Application Migration Service を使用して Amazon EC2 への移行をリホストします。

Migration Portfolio Assessment (MPA)

オンラインツール。これによって、 AWS クラウドに移行するビジネスケースの検証に必要な情報を得られます。MPA は、詳細なポートフォリオ評価 (サーバーの適切なサイジング、価格設定、TCO 比較、移行コスト分析) および移行プラン (アプリケーションデータの分析とデータ収集、アプリケーションのグループ化、移行の優先順位付け、およびウェーブプランニング) を提供します。MPA ツール (ログインが必要) は、すべての AWS コンサルタントと APN パートナーコンサルタントが無料で利用できます。

移行準備状況評価 (MRA)

AWS CAF を使用して、組織のクラウド準備状況に関するインサイトを取得し、長所と短所を特定し、特定されたギャップを埋めるためのアクションプランを構築するプロセス。詳細については、移行準備状況ガイドを参照してください。MRA は、AWS 移行戦略の第一段階です。

移行戦略

ワークロードを AWS クラウドに移行するために使用するアプローチ。詳細については、この用語集の 7 Rs エントリと、「組織を動員して大規模な移行を加速する」を参照してください。

ML

「機械学習」を参照してください。

モダナイゼーション

古い (レガシーまたはモノリシック) アプリケーションとそのインフラストラクチャをクラウド内の俊敏で弾力性のある高可用性システムに変換して、コストを削減し、効率を高め、イノベーションを活用します。詳細については、「AWS クラウドでのアプリケーションのモダナイズ戦略」を参照してください。

モダナイゼーション準備状況評価

組織のアプリケーションのモダナイゼーションの準備状況を判断し、利点、リスク、依存関係を特定し、組織がこれらのアプリケーションの将来の状態をどの程度適切にサポートできるかを決定するのに役立つ評価。評価の結果として、ターゲットアーキテクチャのブループリント、モダナイゼーションプロセスの開発段階とマイルストーンを詳述したロードマップ、特定されたギャップに対処するためのアクションプランが得られます。詳細については、「AWS クラウドでのアプリケーションのモダナイゼーションの準備状況を評価する」を参照してください。

モノリシックアプリケーション (モノリス)

緊密に結合されたプロセスを持つ単一のサービスとして実行されるアプリケーション。モノリシックアプリケーションにはいくつかの欠点があります。1 つのアプリケーション機能エクスペリエンスの需要が急増する場合は、アーキテクチャ全体をスケーリングする必要があります。モノリシックアプリケーションの特徴を追加または改善することは、コードベースが大きくなると複雑になります。これらの問題に対処するには、マイクロサービスアーキテクチャを使用できます。詳細については、「モノリスをマイクロサービスに分解する」を参照してください。

MPA

「Migration Portfolio Assessment」を参照してください。

MQTT

「Message Queuing Telemetry Transport」を参照してください。

多クラス分類

複数のクラスの予測を生成するプロセス (2 つ以上の結果の 1 つを予測します)。例えば、機械学習モデルが、「この製品は書籍、自動車、電話のいずれですか?」 または、「このお客様にとって最も関心のある商品のカテゴリはどれですか?」と聞くかもしれません。

ミュータブルなインフラストラクチャ

本番ワークロードに使用する既存のインフラストラクチャを更新および変更するためのモデル。Well-Architected AWS フレームワークでは、一貫性、信頼性、予測可能性を向上させるために、イミュータブルインフラストラクチャの使用をベストプラクティスとして推奨しています。

O

OAC

「オリジンアクセス制御」を参照してください。

OAI

「オリジンアクセスアイデンティティ」を参照してください。

OCM

「組織変更管理」を参照してください。

オフライン移行

移行プロセス中にソースワークロードを停止させる移行方法。この方法はダウンタイムが長くなるため、通常は重要ではない小規模なワークロードに使用されます。

OI

「オペレーション統合」を参照してください。

Ola

「オペレーショナルレベルアグリーメント」を参照してください。

オンライン移行

ソースワークロードをオフラインにせずにターゲットシステムにコピーする移行方法。ワークロードに接続されているアプリケーションは、移行中も動作し続けることができます。この方法はダウンタイムがゼロから最小限で済むため、通常は重要な本番稼働環境のワークロードに使用されます。

OPC-UA

「Open Process Communications - Unified Architecture」を参照してください。

Open Process Communications - Unified Architecture (OPC-UA)

産業オートメーション用のマシンツーマシン (M2M) 通信プロトコル。OPC-UA により、相互運用の際に、データ暗号化、認証、認可の各スキームを標準化できます。

オペレーショナルレベルアグリーメント (OLA)

サービスレベルアグリーメント (SLA) をサポートするために、どの機能的 IT グループが互いに提供することを約束するかを明確にする契約。

運用準備状況レビュー (ORR)

質問と関連するベストプラクティスのチェックリスト。インシデントや起こり得る障害を理解、評価、防止したり、その範囲を縮小したりする際に役立ちます。詳細については、 AWS Well-Architected フレームワークの「Operational Readiness Reviews (ORR)」を参照してください。

運用テクノロジー (OT)

産業オペレーション、機器、インフラストラクチャを制御するために物理環境と連携させるハードウェアおよびソフトウェアシステム。製造分野では、Industry 4.0 への変革を進める上で、OT と情報技術 (IT) システムの統合に焦点が当てられています。

オペレーション統合 (OI)

クラウドでオペレーションをモダナイズするプロセスには、準備計画、オートメーション、統合が含まれます。詳細については、オペレーション統合ガイドを参照してください。

組織の証跡

組織 AWS アカウント 内のすべての のすべてのイベント AWS CloudTrail を ログに記録する によって作成された証跡 AWS Organizations。証跡は、組織に含まれている各 AWS アカウント に作成され、各アカウントのアクティビティを追跡します。詳細については、CloudTrail ドキュメントの「組織の証跡の作成」を参照してください。

組織変更管理 (OCM)

人材、文化、リーダーシップの観点から、主要な破壊的なビジネス変革を管理するためのフレームワーク。OCM は、変化の導入を加速し、移行問題に対処し、文化や組織の変化を推進することで、組織が新しいシステムと戦略の準備と移行するのを支援します。 AWS 移行戦略では、クラウド導入プロジェクトに必要な変化のスピードにより、このフレームワークは人材アクセラレーションと呼ばれます。詳細については、OCM ガイドを参照してください。

オリジンアクセス制御 (OAC)

Amazon Simple Storage Service (Amazon S3) コンテンツを保護するための、CloudFront のアクセス制限の強化オプション。OAC は AWS リージョン、すべての S3 バケット、 AWS KMS (SSE-KMS) によるサーバー側の暗号化、S3 バケットへの動的 PUT および DELETEリクエストをサポートします。

オリジンアクセスアイデンティティ (OAI)

CloudFront の、Amazon S3 コンテンツを保護するためのアクセス制限オプション。OAI を使用すると、CloudFront が、Amazon S3 に認証可能なプリンシパルを作成します。認証されたプリンシパルは、S3 バケット内のコンテンツに、特定の CloudFront ディストリビューションを介してのみアクセスできます。OAC も併せて参照してください。OAC では、より詳細な、強化されたアクセス制御が可能です。

ORR

「運用準備状況レビュー」を参照してください。

OT

「運用テクノロジー」を参照してください。

アウトバウンド (送信) VPC

AWS マルチアカウントアーキテクチャでは、アプリケーション内から開始されたネットワーク接続を処理する VPC。AWS Security Reference Architecture では、アプリケーションとより広範なインターネット間の双方向のインターフェイスを保護するために、インバウンド、アウトバウンド、インスペクションの各 VPC を使用してネットワークアカウントを設定することを推奨しています。

P

アクセス許可の境界

ユーザーまたはロールが使用できるアクセス許可の上限を設定する、IAM プリンシパルにアタッチされる IAM 管理ポリシー。詳細については、IAM ドキュメントのアクセス許可の境界を参照してください。

個人を特定できる情報 (PII)

直接閲覧した場合、または他の関連データと組み合わせた場合に、個人の身元を合理的に推測するために使用できる情報。PII の例には、氏名、住所、連絡先情報などがあります。

PII

「個人を特定できる情報」を参照してください。

プレイブック

クラウドでのコアオペレーション機能の提供など、移行に関連する作業を取り込む、事前定義された一連のステップ。プレイブックは、スクリプト、自動ランブック、またはお客様のモダナイズされた環境を運用するために必要なプロセスや手順の要約などの形式をとることができます。

PLC

「プログラマブルロジックコントローラー」を参照してください。

PLM

「製品ライフサイクル管理」を参照してください。

ポリシー

次の操作を可能にするオブジェクト: アクセス許可を定義する (ID ベースのポリシーを参照)。アクセス条件を指定する (リソースベースのポリシーを参照)。 AWS Organizations の組織における全アカウントにアクセス許可の上限を定義する (サービスコントロールポリシーを参照)。

多言語の永続性

データアクセスパターンやその他の要件に基づいて、マイクロサービスのデータストレージテクノロジーを個別に選択します。マイクロサービスが同じデータストレージテクノロジーを使用している場合、実装上の問題が発生したり、パフォーマンスが低下する可能性があります。マイクロサービスは、要件に最も適合したデータストアを使用すると、より簡単に実装でき、パフォーマンスとスケーラビリティが向上します。

ポートフォリオ評価

移行を計画するために、アプリケーションポートフォリオの検出、分析、優先順位付けを行うプロセス。詳細については、「移行の準備状況の評価」を参照してください。

述語

`true` または `false` を返すためのクエリ条件。一般的に、`WHERE` 句に記述されます。

述語プッシュダウン

データベースクエリを最適化する手法。これによって、転送前にクエリ内のデータをフィルタリングします。この手法を取ると、リレーショナルデータベースから取得し処理する必要のあるデータの量が減少するため、クエリのパフォーマンスが向上します。

予防的コントロール

イベントの発生を防ぐように設計されたセキュリティコントロール。このコントロールは、ネットワークへの不正アクセスや好ましくない変更を防ぐ最前線の防御です。詳細については、「AWSでのセキュリティコントロールの実装」の「予防的コントロール」を参照してください。

プリンシパル

アクションを実行し AWS 、リソースにアクセスできる のエンティティ。このエンティティは通常、、IAM AWS アカウントロール、または ユーザーのルートユーザーです。詳細については、IAM ドキュメントの「ロールに関する用語と概念」にあるプリンシパルを参照してください。

プライバシーバイデザイン

開発プロセス全体を通してプライバシーが考慮されているシステムエンジニアリングのアプローチ。

プライベートホストゾーン

1 つ以上の VPC 内のドメインとそのサブドメインへの DNS クエリに対し、Amazon Route 53 がどのように応答するかに関する情報を保持するコンテナ。詳細については、Route 53 ドキュメントの「プライベートホストゾーンの使用」を参照してください。

プロアクティブコントロール

非準拠リソースのデプロイ防止を目的としたセキュリティコントロール。このコントロールにより、プロビジョニング前にリソースをスキャンします。コントロールに準拠していないリソースは、プロビジョニングされません。詳細については、 AWS Control Tower ドキュメントの「コントロールリファレンスガイド」および「 セキュリティコントロールの実装」の「プロアクティブコントロール」を参照してください。 AWS

製品ライフサイクル管理 (PLM)

製品の設計、開発、発売から、成長、成熟、衰退、廃棄に至る、製品のライフサイクル全体を通してデータとプロセスを管理すること。

本番環境

「環境」を参照してください。

プログラマブルロジックコントローラー (PLC)

製造分野で使用される、信頼性と適応性に優れたコンピュータであり、これによって、マシンをモニタリングするとともに、製造プロセスを自動化します。

プロンプトチェイニング

1 つの LLM プロンプトによる出力を次のプロンプトの入力に使用して、より良いレスポンスを生成します。この手法を使用すると、複雑なタスクをサブタスクに分割したり、事前レスポンスを繰り返し改良または拡張したりできます。これによって、モデルのレスポンスの精度と関連性が向上し、粒度の高いパーソナライズされた結果を得られます。

仮名化

データセット内の個人識別子をプレースホルダー値に置き換えるプロセス。仮名化は個人のプライバシー保護に役立ちます。仮名化されたデータは、依然として個人データとみなされます。

発行/サブスクライブ (pub/sub)

マイクロサービス間の非同期通信を可能にするパターン。これにより、スケーラビリティと応答性を向上させます。例えば、マイクロサービスベースの MES の場合、マイクロサービスは、他のマイクロサービスがサブスクライブ可能なチャネルにイベントメッセージを発行できます。このシステムでは、発行サービスの変更なしに、新規マイクロサービスを追加できます。

Q

クエリプラン

手順などの一連のステップであり、SQL リレーショナルデータベースシステムのデータにアクセスするために使用されます。

クエリプランのリグレッション

データベースサービスのオプティマイザーが、データベース環境に特定の変更が加えられる前に選択されたプランよりも最適性の低いプランを選択すること。これは、統計、制限事項、環境設定、クエリパラメータのバインディングの変更、およびデータベースエンジンの更新などが原因である可能性があります。

R

RACI マトリックス

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RAG

「検索拡張生成」を参照してください。

ランサムウェア

決済が完了するまでコンピュータシステムまたはデータへのアクセスをブロックするように設計された、悪意のあるソフトウェア。

RASCI マトリックス

「実行責任者、説明責任者、協業先、報告先 (RACI)」を参照してください。

RCAC

「行と列のアクセス制御」を参照してください。

リードレプリカ

読み取り専用に使用されるデータベースのコピー。クエリをリードレプリカにルーティングして、プライマリデータベースへの負荷を軽減できます。

リアーキテクト

「7 Rs」を参照してください。

目標復旧時点 (RPO)

最後のデータリカバリポイントからの最大許容時間です。これにより、最後の回復時点からサービスが中断されるまでの間に許容できるデータ損失の程度が決まります。

目標復旧時間 (RTO)

サービスの中断から復旧までの最大許容遅延時間。

リファクタリング

「7 Rs」を参照してください。

リージョン

地理的エリア内の AWS リソースのコレクション。各 AWS リージョン は、耐障害性、安定性、耐障害性を提供するために、他の から分離され、独立しています。詳細については、「アカウントが使用できる AWS リージョン を指定する」を参照してください。

リグレッション

数値を予測する機械学習手法。例えば、「この家はどれくらいの値段で売れるでしょうか?」という問題を解決するために、機械学習モデルは、線形回帰モデルを使用して、この家に関する既知の事実 (平方フィートなど) に基づいて家の販売価格を予測できます。

リホスト

「7 Rs」を参照してください。

リリース

デプロイプロセスで、変更を本番環境に昇格させること。

再配置

「7 Rs」を参照してください。

リプラットフォーム

「7 Rs」を参照してください。

再購入

「7 Rs」を参照してください。

回復性

中断に抵抗または中断から回復するアプリケーションの機能。 AWS クラウドでの回復力を計画する際には、一般的に、高可用性とディザスタリカバリが考慮されます。詳細については、「AWS クラウド の耐障害性」を参照してください。

リソースベースのポリシー

Amazon S3 バケット、エンドポイント、暗号化キーなどのリソースにアタッチされたポリシー。このタイプのポリシーは、アクセスが許可されているプリンシパル、サポートされているアクション、その他の満たすべき条件を指定します。

実行責任者、説明責任者、協業先、報告先 (RACI) に基づくマトリックス

移行活動とクラウド運用に関わるすべての関係者の役割と責任を定義したマトリックス。マトリックスの名前は、マトリックスで定義されている責任の種類、すなわち責任 (R) 、説明責任 (A) 、協議 (C) 、情報提供 (I) に由来します。サポート (S) タイプはオプションです。サポートが含まれる場合は RASCI マトリックスと呼ばれ、含まれない場合は RACI マトリックスと呼ばれます。

レスポンシブコントロール

有害事象やセキュリティベースラインからの逸脱について、修復を促すように設計されたセキュリティコントロール。詳細については、「 AWSでのセキュリティコントロールの実装」の「レスポンシブコントロール」を参照してください。

保持

「7 Rs」を参照してください。

廃止

「7 Rs」を参照してください。

検索拡張生成 (RAG)

生成 AI の技術。これにより、LLM では、レスポンスの生成前に、トレーニングデータソースの外部にある信頼できるデータソースが参照されます。例えば、RAG モデルによって、組織のナレッジベースまたはカスタムデータのセマンティック検索を実行できる場合があります。細については、「RAG (検索拡張生成) とは何ですか?」を参照してください。

ローテーション

定期的にシークレット情報を更新して、攻撃者が認証情報にアクセスするのをより困難にするプロセス。

行と列のアクセス制御 (RCAC)

アクセスルールが定義された、基本的で柔軟な SQL 表現の使用。RCAC は行権限と列マスクで構成されています。

RPO

「目標復旧時点」を参照してください。

RTO

「目標復旧時間」を参照してください。

ランブック

特定のタスクを実行するために必要な手動または自動化された一連の手順。これらは通常、エラー率の高い反復操作や手順を合理化するために構築されています。

S

SAML 2.0

多くの ID プロバイダー (IdP) が使用しているオープンスタンダード。この機能を使用すると、フェデレーティッドシングルサインオン (SSO) が有効になるため、ユーザーは組織内のすべてのユーザーを IAM で作成しなくても、 AWS マネジメントコンソール にログインしたり AWS 、API オペレーションを呼び出すことができます。SAML 2.0 ベースのフェデレーションの詳細については、IAM ドキュメントの「SAML 2.0 ベースのフェデレーションについて」を参照してください。

SCADA

「監視制御とデータ取得」を参照してください。

SCP

「サービスコントロールポリシー」を参照してください。

シークレット

暗号化された形式で保存するパスワードやユーザー認証情報などの AWS Secrets Manager機密情報または制限付き情報。シークレット値とそのメタデータで構成されます。シークレット値には、バイナリ、1 つの文字列、複数の文字列を指定できます。詳細については、Secrets Manager ドキュメントの「Secrets Manager シークレットの概要」を参照してください。

セキュリティバイデザイン

開発プロセス全体を通してセキュリティが考慮されているシステムエンジニアリングのアプローチ。

セキュリティコントロール

脅威アクターによるセキュリティ脆弱性の悪用を防止、検出、軽減するための、技術上または管理上のガードレール。セキュリティコントロールには、主に 4 つの種類があります。4 つとは、予防、検出、レスポンシブ、プロアクティブです。

セキュリティ強化

アタックサーフェスを狭めて攻撃への耐性を高めるプロセス。このプロセスには、不要になったリソースの削除、最小特権を付与するセキュリティのベストプラクティスの実装、設定ファイル内の不要な機能の無効化、といったアクションが含まれています。

Security Information and Event Management (SIEM) システム

セキュリティ情報管理 (SIM) とセキュリティイベント管理 (SEM) のシステムを組み合わせたツールとサービス。SIEM システムは、サーバー、ネットワーク、デバイス、その他ソースからデータを収集、モニタリング、分析して、脅威やセキュリティ違反を検出し、アラートを発信します。

セキュリティレスポンスの自動化

セキュリティイベントへの自動レスポンスまたは自動修復を目的として、事前定義およびプログラムされたアクション。これらの自動化は、セキュリティのベストプラクティスを実装するのに役立つ検出的または応答的な AWS セキュリティコントロールとして機能します。自動レスポンスアクションの例には、VPC セキュリティグループの変更、Amazon EC2 インスタンスへのパッチ適用、認証情報の更新などがあります。

サーバー側の暗号化

送信先にあるデータの、それ AWS のサービス を受け取る による暗号化。

サービスコントロールポリシー (SCP)

AWS Organizationsの組織内の、すべてのアカウントのアクセス許可を一元的に管理するポリシー。SCP は、管理者がユーザーまたはロールに委任するアクションに、ガードレールを定義したり、アクションの制限を設定したりします。SCP は、許可リストまたは拒否リストとして、許可または禁止するサービスやアクションを指定する際に使用できます。詳細については、 AWS Organizations ドキュメントの「サービスコントロールポリシー」を参照してください。

サービスエンドポイント

のエントリポイントの URL AWS のサービス。ターゲットサービスにプログラムで接続するには、エンドポイントを使用します。詳細については、「AWS 全般のリファレンス」の「AWS のサービス エンドポイント」を参照してください。

サービスレベルアグリーメント (SLA)

サービスのアップタイムやパフォーマンスなど、IT チームがお客様に提供すると約束したものを明示した合意書。

サービスレベルインジケータ (SLI)

エラー率、可用性、スループットといった、サービスパフォーマンス面の指標。

サービスレベル目標 (SLO)

サービスレベルインジケータによって測定され、サービスの状態を表すターゲットメトリクス。

責任共有モデル

クラウドのセキュリティとコンプライアンス AWS について と共有する責任を説明するモデル。AWS はクラウドのセキュリティを担当しますが、お客様はクラウドのセキュリティを担当します。詳細については、「責任共有モデル」を参照してください。

SIEM

「Security Information and Event Management システム」を参照してください。

単一障害点 (SPOF)

特定のアプリケーションを構成する単一の重要なコンポーネントで発生し、システム稼働に支障をきたす可能性のある障害。

SLA

「サービスレベルアグリーメント」を参照してください。

SLI

「サービスレベルインジケータ」を参照してください。

SLO

「サービスレベルの目標」を参照してください。

スプリットアンドシードモデル

モダナイゼーションプロジェクトのスケーリングと加速のためのパターン。新機能と製品リリースが定義されると、コアチームは解放されて新しい製品チームを作成します。これにより、お客様の組織の能力とサービスの拡張、デベロッパーの生産性の向上、迅速なイノベーションのサポートに役立ちます。詳細については、「AWS クラウドでのアプリケーションをモダナイズするための段階的アプローチ」を参照してください。

SPOF

「単一障害点」を参照してください。

スタースキーマ

データベースの編成構造を意味し、1 つの大きいファクトテーブルにトランザクションデータまたは測定データが保存され、1 つ以上の小さいディメンションテーブルにデータ属性が保存されます。この構造は、データウェアハウスやビジネスインテリジェンスを用途とするように設計されています。

strangler fig パターン

レガシーシステムが廃止されるまで、システム機能を段階的に書き換えて置き換えることにより、モノリシックシステムをモダナイズするアプローチ。このパターンは、宿主の樹木から根を成長させ、最終的にその宿主を包み込み、宿主に取って代わるイチジクのつるを例えています。そのパターンは、モノリシックシステムを書き換えるときのリスクを管理する方法として Martin Fowler により提唱されました。このパターンの適用方法の例については、「コンテナと Amazon API Gateway を使用して、従来の Microsoft ASP.NET (ASMX) ウェブサービスを段階的にモダナイズ」を参照してください。

サブネット

VPC 内の IP アドレスの範囲。サブネットは、1 つのアベイラビリティーゾーンに存在する必要があります。

監視制御とデータ取得 (SCADA)

製造分野において、ハードウェアとソフトウェアを使用して物理アセットと本番運用をモニタリングするシステム。

対称暗号化

データの暗号化と復号に同じキーを使用する暗号化のアルゴリズム。

合成テスト

ユーザーとのやり取りをシミュレートして、起こり得る問題を検出したり、パフォーマンスをモニタリングしたりすることで、システムをテストします。Amazon CloudWatch Synthetics を使用すると、こうしたテストを作成できます。

システムプロンプト

コンテキスト、指示、ガイドラインなどを提示して、LLM に動作を指示する手法。システムプロンプトは、コンテキストを設定して、ユーザーとやり取りするルールを確立するのに有用です。

T

タグ

AWS リソースを整理するためのメタデータとして機能するキーと値のペア。タグは、 リソースの管理、識別、整理、検索、フィルタリングに役立ちます。詳細については、「AWS リソースのタグ付け」を参照してください。

ターゲット変数

監督された機械学習でお客様が予測しようとしている値。これは、結果変数のことも指します。例えば、製造設定では、ターゲット変数が製品の欠陥である可能性があります。

タスクリスト

ランブックの進行状況を追跡するために使用されるツール。タスクリストには、ランブックの概要と完了する必要のある一般的なタスクのリストが含まれています。各一般的なタスクには、推定所要時間、所有者、進捗状況が含まれています。

テスト環境

「環境」を参照してください。

トレーニング

お客様の機械学習モデルに学習するデータを提供すること。トレーニングデータには正しい答えが含まれている必要があります。学習アルゴリズムは入力データ属性をターゲット (お客様が予測したい答え) にマッピングするトレーニングデータのパターンを検出します。これらのパターンをキャプチャする機械学習モデルを出力します。そして、お客様が機械学習モデルを使用して、ターゲットがわからない新しいデータでターゲットを予測できます。

トランジットゲートウェイ

VPC とオンプレミスネットワークを相互接続するために使用できる、ネットワークの中継ハブ。詳細については、 AWS Transit Gateway ドキュメントの「トランジットゲートウェイとは」を参照してください。

トランクベースのワークフロー

デベロッパーが機能ブランチで機能をローカルにビルドしてテストし、その変更をメインブランチにマージするアプローチ。メインブランチはその後、開発環境、本番前環境、本番環境に合わせて順次構築されます。

信頼されたアクセス

ユーザーに代わって AWS Organizations およびそのアカウントで組織内でタスクを実行するために指定したサービスにアクセス許可を付与します。信頼されたサービスは、サービスにリンクされたロールを必要なときに各アカウントに作成し、ユーザーに代わって管理タスクを実行します。詳細については、 ドキュメントの「 を他の AWS のサービス AWS Organizations で使用する AWS Organizations 」を参照してください。

チューニング

機械学習モデルの精度を向上させるために、お客様のトレーニングプロセスの側面を変更する。例えば、お客様が機械学習モデルをトレーニングするには、ラベル付けセットを生成し、ラベルを追加します。これらのステップを、異なる設定で複数回繰り返して、モデルを最適化します。

ツーピザチーム

2 枚のピザを分け合えることができるくらい小さな DevOps チーム。ツーピザチームの規模では、ソフトウェア開発におけるコラボレーションに最適な機会が確保されます。

U

不確実性

予測機械学習モデルの信頼性を損なう可能性がある、不正確、不完全、または未知の情報を指す概念。不確実性には、次の 2 つのタイプがあります。認識論的不確実性は、限られた、不完全なデータによって引き起こされ、弁論的不確実性 は、データに固有のノイズとランダム性によって引き起こされます。

未分化なタスク

ヘビーリフティングとも呼ばれ、アプリケーションの作成と運用には必要だが、エンドユーザーに直接的な価値をもたらさなかったり、競争上の優位性をもたらしたりしない作業です。未分化なタスクの例としては、調達、メンテナンス、キャパシティプランニングなどがあります。

上位環境

「環境」を参照してください。

V

バキューミング

ストレージを再利用してパフォーマンスを向上させるために、増分更新後にクリーンアップを行うデータベースのメンテナンス操作。

バージョンコントロール

リポジトリ内のソースコードへの変更など、変更を追跡するプロセスとツール。

VPC ピアリング

プライベート IP アドレスを使用してトラフィックをルーティングできる、2 つの VPC 間の接続。詳細については、Amazon VPC ドキュメントの「VPC ピア機能とは」を参照してください。

脆弱性

システムのセキュリティを脅かすソフトウェアまたはハードウェアの欠陥。

W

ウォームキャッシュ

頻繁にアクセスされる最新の関連データを含むバッファキャッシュ。データベースインスタンスはバッファキャッシュから、メインメモリまたはディスクからよりも短い時間で読み取りを行うことができます。

ウォームデータ

アクセス頻度の低いデータ。この種類のデータをクエリする場合、通常は適度に遅いクエリでも問題ありません。

ウィンドウ関数

現在のレコードに何らかの形で関連している行のグループに計算を実行する SQL 関数。ウィンドウ関数は、移動平均を計算したり、現在の行の相対位置に基づいて他の行の値にアクセスするといったタスクの処理に役立ちます。

ワークロード

ビジネス価値をもたらすリソースとコード (顧客向けアプリケーションやバックエンドプロセスなど) の総称。

ワークストリーム

特定のタスクセットを担当する移行プロジェクト内の機能グループ。各ワークストリームは独立していますが、プロジェクト内の他のワークストリームをサポートしています。たとえば、ポートフォリオワークストリームは、アプリケーションの優先順位付け、ウェーブ計画、および移行メタデータの収集を担当します。ポートフォリオワークストリームは、これらの設備を移行ワークストリームで実現し、サーバーとアプリケーションを移行します。

WORM

「Write-Once-Read-Many」を参照してください。

WQF

「AWS ワークロード資格フレームワーク」を参照してください

Write-Once-Read-Many (WORM)

データを 1 回のみ書き込むことで、データの削除や変更を防ぐストレージモデル。承認済みユーザーは、必要な回数だけデータを読み取ることができますが、変更することはできません。このデータストレージインフラストラクチャは、イミュータブルと見なされます。

Z

ゼロデイエクスプロイト

ゼロデイ脆弱性を悪用した攻撃 (一般的にマルウェアによる)。

ゼロデイ脆弱性

実稼働システムにおける未解決の欠陥または脆弱性。脅威アクターは、このような脆弱性を利用してシステムを攻撃する可能性があります。開発者は、よく攻撃の結果で脆弱性に気付きます。

ゼロショットプロンプト

LLM にタスク実行の手順は提示するが、実行のガイドとして役立つ例 (ショット) は提示しない方法。LLM は、事前トレーニング済みの知識を使用してタスクを処理する必要があります。ゼロショットプロンプトの有効性は、タスクの複雑さとプロンプトの品質によって異なります。「数ショットプロンプト」も参照してください。

ゾンビアプリケーション

平均 CPU およびメモリ使用率が 5% 未満のアプリケーション。移行プロジェクトでは、これらのアプリケーションを廃止するのが一般的です。