データベース

データベース概要

いつものDB

リレーショナルデータベース

ビッグデータ向け

DynamoDB

リレーショナルデータベース

データ間の関係性が定義されたデータを取り扱うDBシステム

ER図などを使って表現

利用データ：会計データ顧客データなどの構造化されたシステム

NoSQL

非構造化：テキスト・動画・音声

半構造化：XML・JSON・文書

リレーショナル構造を持たない

構造化されていないKeyとValueのみのKVSデータ

IoTと同様にビッグデータ解析に利用する

タイプ：

キーバリューストア：

高速な読み込み・分散型拡張に優れる

ワイドカラムストア：

追加で無限にカラムを追加できる→複雑な管理が可能

ドキュメントデータベース：

JSONやXMLなどのデータを格納

複雑なデータ構造を扱うアプリで生産性高く柔軟に開発する

グラフデータベース：

RDBと比較して高速横断検索が可能

グラフ理論に基づいたデータ同士の関係性をグラフでむすびつく

DynamoDB

プロビジョンドスループット

プロビジョンド（準備）をする

DBのクラッドに応じてキャパシティを割り当てられる

ストレージ容量の制限がない

整合性モデル

結果整合性モデル

オプションで一部のみ強い整合性が実現可能

パーティショニング

テーブルＡのパーティションを３つのＡＺに振り分けて並列処理

ユースケース

ビッグデータ・・・大量データの収集・蓄積・分析が可能

アプリケーション・・・大規模サービスでのデータ高速処理が必要な

アプリケーション向けに活用

多数のユーザがアクセスする

ユーザ行動データ管理・・・ユーザ情報やゲーム、広告などの行動履歴管理

バックエンドデータ処理・・・ログ管理、バッチ処理のロック管理

ストレージのインデックス

適用分析

ＩｏＴデータの収集・・・DynamoDB

ゲームの行動記録　・・・DynamoDB

銀行の振込処理　　・・・RDS

テーブル設計

テーブル　　　・・・データのコレクション

　　　　RDSと同様

項目(アイテム)・・・ 例：個人情報

属性　　　　　・・・例：個人情報（ID）　個人情報(名前)

インデックス

暗黙的なキー・・・ ハッシュキーやレンジキーを宣言して検索に利用する

ハッシュキー・・・主キー（必須）

レンジキー　・・・副次キー

明示的なキー・・・ ローカルセカンダリインデックス（LSI）

レンジキーをもうひとつ作るイメージ

１テーブルに５つ作成可能

テーブル作成後に作成

グローバルセカンダリインデックス（GSI）

全データに対して検索をかける場合などに使用

１テーブルに５つ作成可能

テーブル作成後に作成

テーブル操作

GetItem・・・ハッシュキーを条件に一定の項目を取得

PutItem・・・1件書き込み

Update・・・1件更新

Delete・・・1kwン削除

Query・・・ハッシュキーとレンジキーにマッチする項目を取得

Scan・・・テーブルを全件検索

BatchGet・・・複数のプライマリーキーに対してマッチする項目取得

DynamoDB　Streams：イベント発生時の履歴をキャプチャできる機能

データの保存・・・過去24時間以内のデータ変更の履歴を保存委24時間を経過すると消去される

データの容量はマネージド型で自動的に管理

データ保存の順番・・・操作が実施された順番に応じてデータはシリアライズ

特定のハッシュキーに基づいた変更は正しい順番で保存されるが、ハッシュキーが異なる場合は受信した順番が前後される可能性がある

ユースケース

クロスリージョンレプリケーション

データ更新をトリガーとしたアプリケーション

DynamoDB→Lambda→ DynamoDBの自動更新

S3にログ保存

携帯電話などにプッシュ通知

DAX（DynamoDB　Accelerator）

インメモリキャッシュ型の機能を付けて高速化する

DynamoDB　→　DAXクラスター（キャッシュ）→　EC2

導入が容易

ミリ行単位の結果整合性が欲しい

ユースケース：トレーディングシステム

プロビジョニングをあまりしなくてよいためコスト最適化にも貢献

グローバルテーブル

リージョン間でレプリケーションされるマルチマスターテーブルを作成可能

オンデマンドバックアップ

数百ＴＢのバックアップを実行可能

長期期間用のバックアップ可能

DynamoDB活用のコツ

既存のRDSをLambdaなどと組み合わせたりできないか？検証する

アーキテクチャ設計を見直す一番に考えてみてもよい

Aurora

Auroraの概要

クラウドに適した分散型RDS

NoSQLのような処理をRDBで実現している

クラウド専門DB

MYSQLの2.5～5倍の性能

商用DBの1/10の価格で利用可能

AWS側がAuroraを推奨している

Auroraの特徴

高い並列処理が可能

大量の読み書きに優れる

MYSQL5.6互換を選択可能

PostgreSQL互換を選択可能

→Aurora移行が検討できる

耐障害性/自己回復性

３つのAZに２つのコピー：６つの分散ができ仮想ボリューム化できる

99.99％の高可用性と耐久性のフルマネージドサービス

スケーラビリティ

AutoScaling使用可能

リードレプリカ１５個配置可能

DBクラスタ構成

Aurora　Writer（マスタ）

Aurora　Reader（リードレプリカ）

→　エンドポイント

EC2から書き込み処理→マスタが処理

EC2から読み込み処理→リードレプリカが処理

マスタ止まったらリードレプリカがフェイルオーバー

マイグレーション

MYSQL　→　スナップショット　→　Aurora with MYSQL

マルチマスタ構成

マスタを別AZに配置可能且つWrite機能もスケーリングできる

Auroraサーバレス

予測困難なアプリケーションワークロードに対応したオンデマンド自動

スケーリング構成

AuroraグローバルDB

他のリージョンに対する高性能リードレプリカ作成

低レイテンシー

Auroraユースケース

大規模なクエリデータ処理

テーブル数多い

コネクション多い

書き込み多くトランザクション量が多い

運用の容易さを活用する

Auroraサーバレス

Auroraサーバレスモードで構築可能

予測不能なワークロードやアクティビティが発生しない時が多い

→　サーバレス

インスタンスクラスを選択しない：ｔシリーズやｒシリーズなど

代わりにＡＣＵ（AuroraCapacityUnit）という範囲で決定

キャパシティを最小と最大値で設定する

AutoScalingのタイムアウト時の行動を選択

アイドル状態の挙動を決められる

ストレージは可変しない

実働時間が予測不能で変動が激しいケースではコストが最も安くなる！！！！

可用性や処理性能などはAuroraのが上

EFS

EFS概要

ＮＡＳに似たファイルストレージ

ファイルシステムとして利用する

複数のEC2インスタンスでの共有アクセスが可能

特徴

シンプルでスケーラブルなフルマネージド型サービス

ペタバイトまでスケーラブルにデータを蓄積

ファイルの減少に合わせて拡張縮小を自動で行う

基本性能

何千もの同時アクセスが可能

ファイル名は２５５バイト

複数ＡＺにまたがった構成

設定

ファイルシステム作成

ＥＦＳの管理単位

ファイルやディレクトリの保管場所

マウントターゲット

ＶＰＣ内のＡＺの接続先

固定のＤＮＳ名とＩＰアドレスを有している

ＥＣ２　→　マウントターゲット　→　ＥＦＳ

パフォーマンスモード

汎用モード

一般的な用途を想定

最大ＩＯモード

何十から何千の同時アクセスを想定

レイテンシーが多少ある

バースト機能

スケーラビリティ確保

一時的にサーバのパフォーマンスを向上：設定した上限を超えて使用

ＥＦＳクライアント

ＥＦＳマウントヘルパー：Amazonから提供

Linux NFSv4クライアント

利用方針

複数からの同時アクセスが必要

利用シーン

コンテンツの共有リポジトリ

アプリケーションの共有ディレクトリ

ビッグデータなどの分散並列処理環境における共有データアクセスストレージ

３つのファイルストレージ

EFS

Amazom FSx For WindowsFile Server

Amazon FSx For Lustre　・・・　機械学習などの小ストレージ

Amazom FSx For WindowsFile Server

アクティブディレクトリ統合などの広い管理機能

SMBプロトコルにより幅広く接続可能

アーキテクチャ構成

ENI経由でアクセス・・・VPCのセキュリティグループでトラフィック制御可能

マルチAZ構成可能

他AZ内からのアクセス可能

Amazon FSx For Lustre

スパコン向けの高性能なものに向ける

最適容量3600GBスループット

最速！！！！

増大するデータ量への対応

IoTやビッグデータなどで絶えず増加しておくデータ量への対応

対応

①効率的なデータ量への対応

S3 Glacier Glue

②ストリームデータ処理

Kinesis (IoTには必須)

③大量データの解析手法

EMR　Athena　Redshift　QuickSight

ビッグデータに必要な技術

大量データ　　 大量データの蓄積技術

多様なデータ 多様データの蓄積技術

速い処理 速い処理が可能なソフトウェア

大量データバッチ処理向け

Apache　Hadoop

ストリーミング処理向け

Apache　Spark

Kinesis

Kinesisの概要

AmazomKinesisDataStreams

ストリームデータ処理

AmazomKinesisDataFirehose

S3などに配信

AmazomKinesisDataAnalytics

SQLクエリで分析

IoTデータを取得するサービスと連携

ストリーム

Shard1

Shard2

Shard3・・・処理性能を増やしていく

AmazomKinesisDataStreams

データ提供側（プロデューサ）

AmazomKinesisDataStreams

利用側（コンシューマ）

AmazomKinesisAgent

アプリケーションに組み込んでデータを取得

AmazomKinesisProducerLibrary（KPL）

KinesisStreamsにデータを送信するOSSの補助ライブラリ

Fuluent plugin for AmazomKinesis

KinesisStreamsとKinesisFirehoseにイベントを送信するFluentd出力プラグイン

AmazomKinesisDataGenerator（KDG）

KinesisStreamsとKinesisFirehoseにテストデータを簡単に送信する

AmazomKinesisClientLibrary（KCL）

OSSのクライアントライブラリEC2インスタンスなどにデプロイして使う

Kinesisアプリケーションを作成することができる

AWS Active Directory

ユーザ管理サービス（ＩＡＭ以外）

AWS Directory Service

AWS上でActiveDirectoryを利用してActiveDirectoryによるユーザ管理を行う

オンプレミス環境のADとの連携やシングルサインオンを実現するサービス

Amazon Cognito

Webアプリケーションやモバイルアプリケーションにユーザのサインアップや

サインインおよびアクセスコントロール機能を提供する

AWS Single Sign-on（SSO）

複数のAWSアカウントとビジネスアプリケーションへのアクセスの一元的な

管理を容易にシングルサインオンアクセスをユーザに提供するサービス

AWS Security Token Service(STS)

ＩＡＭユーザまたはＡＤによって認証されたフェデレーションユーザに対して

一時的な制限付き特権の認証情報をリクエストを可能にするサービス

ActiveDirectory

ユーザ名とパスワードによってユーザを認証する仕組み

管理するユーザ情報

ID・ユーザ名・メールアドレス・パスワードなど

実現する機能

運用効率の向上

コンプライアンス向上

セキュリティの強化

アプリのアクセス制御

ファイル共有

パッチ管理

AWS Active Directory Serviceの選択

Simple　AD

フルマネージド型のディレクトリを新規作成

スモール：500ユーザ　ラージ：5000ユーザをサポート

AWS Managed Microsoft ADを機能サブセット使用可能

外部ユーザにAWSリソースを提供

AD　Connector

オンプレミス側のADの認証情報を利用してAWSのリソースにアクセス

AWS Managed Microsoft AD

オンプレミスのADとは信頼関係を設定してSSOを実現

SimpleAD

VPC内に構成

AWSで提供しているサービスに外部ユーザがアクセスできる

→IAMユーザ以外でも簡易にAmazomWorkSpaceに認証アクセスが可能に！

ADConnector

既存のDirectoryを活かしてAWS環境へのアクセスを可能にするサービス

AWS Managed Microsoft AD

AWS上にMicrosoftADを作成できる

オンプレミス環境とAWSユーザ管理を統合管理するサービス

AWS　SSOの利用

SSOアクセスの一元管理を容易にするSSOサービス

SAML2.0を使用してIDフェデレーションをサポート

オンプレミス側からAWSコンソールにアクセス可能

AWS　STS

一時的なセキュリティ認証情報を提供するサービス

Cognito

アプリケーションにユーザ認証機能を付与したい場合に利用する