AWS Well-Architected Framework

パフォーマンス効率の柱

パフォーマンス効率の柱: AWS Well-Architected Framework

Table of Contents

パフォーマンス効率の柱 - AWS Well-Architected フレームワーク

公開日: 2024 年 6 月 27 日 (改訂履歴)

要約

このホワイトペーパーは、 AWS Well-Architected フレームワークを確認してください。このドキュメントの目的は、お客様がクラウドリソースを効率的に使用してビジネス要件を満たし、需要の変化やテクノロジーの進化に応じてその効率を維持するために役立つガイダンスを提供することです。

はじめに

それらの AWS Well-Architected Framework は、AWS でワークロードを構築する際に行う決定のメリットとデメリットを理解するのに役立ちます。このフレームワークを利用すると、信頼性および有効性が高く、セキュアでコスト効率に優れた持続可能なワークロードをクラウド上で設計および運用するための、アーキテクチャに関するベストプラクティスを学ぶことができます。このフレームワークは、アーキテクチャをベストプラクティスに照らし合わせて一貫的に測定し、改善すべき領域を特定する手段を提供します。Well-Architected ワークロードを備えることによって、ビジネス成功の可能性が大幅に高まると私たちは確信しています。

このフレームワークは次の 6 つの柱に基づいています。

- オペレーショナルエクセレンス
- セキュリティ
- 信頼性
- パフォーマンス効率
- コスト最適化
- サステナビリティ

本書では、パフォーマンス効率の柱の原則をお客様のワークロードに適用する方法について説明します。従来のオンプレミス環境で、持続する優れたパフォーマンスを達成することは容易ではありません。本書で説明する原則は、長期にわたって持続的なパフォーマンスを効率的に提供するアーキテクチャを AWS で構築するために役立ちます。このドキュメントのガイダンスとベストプラクティス

は、AWS でパフォーマンス効率の高いクラウドソリューションを構築するための指針となる 5 つの主要な重点分野を対象としています。重点分野は次のとおりです。

- アーキテクチャの選択
- コンピューティングとハードウェア
- データ管理
- ネットワークとコンテンツ配信
- プロセスと文化

このホワイトペーパーの対象者は、最高技術責任者 (CTO)、アーキテクト、デベロッパー、オペレーションチームメンバーなどの技術担当者です。このホワイトペーパーを読むことで、高性能なクラウドアーキテクチャを設計し運用する際に使われる AWS のベストプラクティスと戦略を理解できます。

パフォーマンス効率

パフォーマンス効率の柱では、要件を満たすためのコンピューティングリソースの効率的な使用、および需要の変化と技術の進化に合わせて効率性を維持する方法に焦点を当てています。

トピック

設計原則

以下の設計原則は、クラウド内で効率的なワークロードを実現し、維持するために役立ちます。

- 最新テクノロジーを誰もが利用できるようにする: 複雑なタスクをクラウドベンダーに委託することによって、チームがより簡単に高度なテクノロジーを実装できるようにします。IT チームに新しいテクノロジーのホストと実行について学んでもらうのではなく、テクノロジーをサービスとして消費することを検討します。たとえば、NoSQL データベース、メディアトランスコーディング、および機械学習などは、いずれも特化された専門知識を必要とするテクノロジーです。クラウドでは、これらのテクノロジーがチームによる消費が可能なサービスとなり、チームはリソースのプロビジョニングと管理ではなく、製品の開発に集中できるようになります。
- わずか数分でグローバル展開する: 世界中の複数の AWS リージョンにワークロードをデプロイすることで、最小限のコストで、より低いレイテンシーとより優れたエクスペリエンスを提供できます。
- サーバーレスアーキテクチャを使用する: サーバーレスアーキテクチャは、従来のコンピューティングアクティビティのために物理的なサーバーを実行して維持する必要性を取り除きます。たとえば、サーバーレスストレージサービスは静的ウェブサイトとして機能させることができ (ウェブサイトサーバーが不要になる)、イベントサービスはコードをホストできます。これによって物理サーバーを管理する運用上の負担が取り除かれます。また、マネージドサービスはクラウド規模で運用されることから、トランザクションコストも削減することができます。
- より頻繁に実験する: 仮想的で自動化できるリソースを使うことで、異なるタイプのインスタンス、ストレージ、設定を使用した比較テストを簡単に実施できます。
- メカニカルシンパシーを重視する: 目標に最適なテクノロジーアプローチを使用します。例えば、ワークロードで使用するデータベースやストレージを選択するときには、データアクセスのパターンを考慮します。

定義

クラウドでパフォーマンス効率を実現するには、次の領域に焦点を合わせます。

- アーキテクチャの選択
- コンピューティングとハードウェア
- データ管理
- ネットワークとコンテンツ配信
- プロセスと文化

データ駆動型アプローチを採#して、#パフォーマンスのアーキテクチャを構築します。高レベルの設計からリソースタイプの選択と設定まで、アーキテクチャのすべての側面におけるデータを収集します。

選択した内容を定期的に見直して、進化し続ける AWS クラウドの機能を十分に活かしていることを確認します。モニタリングすることで、期待されるパフォーマンスからの逸脱を確実に把握できるようにします。さらに、パフォーマンスを向上させるために、圧縮やキャッシングの使用、または整合性要件の緩和などのアーキテクチャ面でのトレードオフを行います。

アーキテクチャの選択

特定のワークロードに適したソリューションはさまざまで、大抵の場合、ソリューションには複数のアプローチが組み合わされています。優れた設計のワークロードは、パフォーマンスを向上させるために複数のソリューションを使用し、異なる機能を有効化します。

AWS のリソースはあらゆるタイプと設定で利用できるため、お客様のニーズを満たす、最も近いアプローチを簡単に見つけることができます。また、オンプレミスのインフラストラクチャでは簡単に実現できないオプションも利用できます。例えば、Amazon DynamoDB のようなマネージドサービスでは、あらゆるスケールにおいてレイテンシーが 10 ミリ秒未満であるフルマネージド型の NoSQL データベースを提供します。

この重点分野では、効率的で高性能なクラウドリソースとアーキテクチャパターンを選択する方法に関するガイダンスとベストプラクティスを紹介します。

ベストプラクティス

- PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する
- PERF01-BP02 クラウドプロバイダーまたは適切なパートナーからのガイダンスを使用して、アーキテクチャパターンとベストプラクティスについて学ぶ
- PERF01-BP03 アーキテクチャに関する意思決定においてコストを考慮する
- PERF01-BP04 トレードオフが顧客とアーキテクチャの効率にどのように影響するかを評価する
- PERF01-BP05 ポリシーとリファレンスアーキテクチャを使用する
- PERF01-BP06 ベンチマークを使用してアーキテクチャに関する意思決定を行う
- PERF01-BP07 データ駆動型のアプローチでアーキテクチャを選択する

PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する

利用可能なサービスや設定について継続的に学び、発見することで、アーキテクチャに関する意思決定をより適切に行い、ワークロードアーキテクチャのパフォーマンス効率を向上させることができます。

一般的なアンチパターン:

- クラウドをコロケーションされたデータセンターとして使用する。

- クラウドへの移行後、アプリケーションをモダナイズしない。
- 永続化する必要があるすべてのものに対して、1 つのストレージタイプのみを使用する。
- 現在の基準に最も近いインスタンスタイプを使用するが、必要に応じてより大きいインスタンスタイプを使用する。
- マネージドサービスとして使用できるテクノロジーをデプロイおよび管理する。

このベストプラクティスを活用するメリット: 新しいサービスと設定を検討することで、パフォーマンスを大幅に向上させ、コストを削減し、ワークロードの維持に必要な労力を最適化できる場合があります。また、クラウド対応製品の価値実現までの時間を短縮できる可能性もあります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

AWS では、パフォーマンスを向上させ、クラウドワークロードのコストを削減するための新しいサービスや機能を継続的にリリースしています。クラウドでのパフォーマンスの有効性を維持するためには、こうした新しいサービスや機能に関する最新情報を常に把握しておくことが重要です。ワークロードアーキテクチャのモダナイズは、生産性の向上、イノベーションの促進、成長機会の拡大にも役立ちます。

実装手順

- 関連サービスのワークロードソフトウェアとアーキテクチャを棚卸しします。どのカテゴリの製品について詳しく調べるかを決めます。
- AWS の提供サービスを調べ、パフォーマンスの向上、コストの削減、運用の煩雑さの軽減に役立つ関連サービスと設定オプションを特定、把握します。
  - Amazon Web Services クラウド
  - AWS Academy
  - AWS の最新情報
  - AWS ブログ
  - AWS Skill Builder
  - AWS イベントスケジュール
  - AWS トレーニング と認定
  - AWS YouTube チャンネル
  - AWS ワークショップ

  - AWS コミュニティ
- サンドボックス (非実稼働) 環境を使用して、追加コストをかけずに新しいサービスについて学び、試してみます。
- 新しいクラウドサービスや機能について継続的に学びましょう。

リソース

関連するドキュメント:

- Overview of Amazon Web Services
- Amazon EC2 の特徴
- AWS パートナーラーニングプランでステップバイステップで学ぶ
- AWS トレーニングと認定
- My learning path to become an AWS solutions architect
- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター
- AWS でモダンアプリケーションを構築する

関連動画:

- AWS re:Invent 2023 - What's new with Amazon EC2
- AWS re:Invent 2022 - Reduce your operational and infrastructure costs with Amazon ECS
- AWS re:Invent 2023 - Build with the efficiency, agility & innovation of the cloud with AWS
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- This is my Architecture

関連サンプル:

- AWS Samples
- AWS SDK サンプル

PERF01-BP02 クラウドプロバイダーまたは適切なパートナーからのガイダンスを使用して、アーキテクチャパターンとベストプラクティスについて学ぶ

アーキテクチャに関する意思決定の指針として、ドキュメント、ソリューションアーキテクト、プロフェッショナルサービス、適切なパートナーなどのクラウド企業のリソースを活用します。こうしたリソースを利用することで、アーキテクチャを評価、改善し、最適なパフォーマンスを実現できます。

一般的なアンチパターン:

- AWS を一般的なクラウドプロバイダーとして使用する。
- 意図されていない方法で AWS のサービスを利用する。
- ビジネス上の背景を考慮せずに、すべてのガイダンスに従う。

このベストプラクティスを活用するメリット: クラウドプロバイダーや適切なパートナーからのガイダンスを利用することで、ワークロードに適したアーキテクチャを選択し、自信を持って意思決定を行うことができます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

AWS には、効率的なクラウドワークロードの構築と管理に役立つさまざまなガイダンス、ドキュメント、リソースが用意されています。AWS ドキュメントには、コードサンプル、チュートリアルのほか、サービスについての詳細な説明が記載されています。ドキュメントの他にも、AWS では、お客様がクラウドサービスのさまざまな側面を探求し、AWS で効率的なクラウドアーキテクチャを実装するのに役立つトレーニングおよび認定プログラム、ソリューションアーキテクト、プロフェッショナルサービスを提供しています。

こうしたリソースを活用して、貴重な知識やベストプラクティスに関するインサイトを導き出し、AWS クラウド での時間の節約、成果の向上につなげましょう。

実装手順

- AWS ドキュメントとガイダンスを確認し、ベストプラクティスに従ってください。こうしたリソースは、サービスの効果的な選定および設定、パフォーマンスの向上に役立ちます。

  - AWS ドキュメント (ユーザーガイドやホワイトペーパーなど)
  - AWS ブログ
  - AWS トレーニング と認定
  - AWS YouTube チャンネル
- AWS パートナーイベント (AWS Global Summits、AWS re:Invent、ユーザーグループ、ワークショップなど) に参加して、AWS のエキスパートから AWS のサービスを利用する際のベストプラクティスを学びましょう。
  - AWS パートナーラーニングプランでステップバイステップで学ぶ
  - AWS イベントスケジュール
  - AWS ワークショップ
  - AWS コミュニティ
- 追加のガイダンス、または製品情報が必要な場合は、AWS までお問い合わせください。AWS ソリューションアーキテクトおよび AWS プロフェッショナルサービス は、ソリューションの実装に関するガイダンスを提供します。 AWS パートナー は、ビジネスの俊敏性とイノベーションを引き出すために AWS の専門知識を提供します。
- サービスを効果的に使用するためにテクニカルサポートが必要な場合は、 AWS Support を使用します。 AWS サポートプランは、 お客様がパフォーマンスを最適化し、リスクとコストを管理しながら AWS をうまく活用できるように、適切なツールと専門知識へのアクセスを提供するように設計されています。

リソース

関連するドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター
- AWS エンタープライズサポート

関連動画:

- This is my Architecture

- AWS re:Invent 2023 - Advanced event-driven patterns with Amazon EventBridge
- AWS re:Invent 2023 - Implementing distributed design patterns on AWS
- AWS re:Invent 2023 - Application architecture as code

関連する例:

- AWS Samples
- AWS SDK サンプル
- AWS Analytics Reference Architecture

PERF01-BP03 アーキテクチャに関する意思決定においてコストを考慮する

アーキテクチャに関する意思決定でコストを考慮すると、クラウドワークロードのリソース使用率とパフォーマンス効率が向上します。クラウドワークロードがコストに及ぼす影響を認識していれば、効率的なリソースを活用し、無駄な作業を減らせる可能性が高くなります。

一般的なアンチパターン:

- インスタンスの 1 つのファミリーのみを使用する。
- ライセンスソリューションとオープンソースソリューションを比較しない。
- ストレージライフサイクルポリシーを定義しない。
- AWS クラウド の新しいサービスや機能を確認しない。
- あなたは、ブロックストレージのみを使用します。

このベストプラクティスを活用するメリット: 意思決定にコストを考慮することで、より効率的なリソースを使用し、他の投資を検討できるようになります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

コストを考慮してワークロードを最適化することで、リソース使用率を向上させ、クラウドワークロードの無駄を省くことができます。アーキテクチャ上の意思決定にコストを考慮に入れることに

は、通常、ワークロードコンポーネントのライトサイジングと伸縮性の有効化が含まれます。これにより、クラウドワークロードのパフォーマンス効率が向上します。

実装手順

- クラウドワークロードの予算制限などのコスト目標を設定します。
- ワークロードのコストを左右する主要コンポーネント (インスタンスやストレージなど) を特定します。AWS Pricing Calculator お および AWS Cost Explorer を使用して、 ワークロードの主なコスト要因を特定できます。
- クラウドの 料金モデル を理解します (オンデマンド、リザーブドインスタンス、Savings Plans、スポットインスタンスなど)。
- Well-Architected コスト最適化のベストプラクティス を使用して、主なコスト要因を最適化します。
- コストを継続的にモニタリングおよび分析して、ワークロードにおけるコスト最適化の機会を特定します。
  - AWS Budgets を使用して、 許容できないコストに関するアラートを受け取ります。
  - AWS Compute Optimizer ま たは AWS Trusted Advisor を使用して、コスト最適化に関する推奨事項を確認します。
  - AWS Cost Anomaly Detection を使用して、異常なコストの検出と根本原因の分析を自動化します。

リソース

関連するドキュメント:

- What is AWS Billing and Cost Management?
- AWS によるコスト最適化
- Choosing an AWS cost management strategy
- A Beginner's Guide to AWS Cost Management
- A Detailed Overview of the Cost Intelligence Dashboard
- AWS アーキテクチャセンター
- AWS ソリューションライブラリ
- AWS ナレッジセンター

関連動画:

- This is my Architecture
- AWS re:Invent 2023 - What's new with AWS cost optimization
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2023 - Optimize costs in your multi-account environments

関連する例:

- AWS Compute Optimizer デモコード
- コスト最適化ワークショップ
- Cloud Financial Management Technical Implementation Playbooks
- Startup optimization: Tuning application performance for maximum efficiency
- Serverless Optimization Workshop (Performance and Cost)
- Scaling cost effective architectures

PERF01-BP04 トレードオフが顧客とアーキテクチャの効率にどのように影響するかを評価する

パフォーマンス関連の改善を評価する際には、どの選択肢が顧客、そしてワークロードの効率性に影響するかを特定します。例えば、key-value データストアを使用することでシステムパフォーマンスが向上する場合、その結果整合性の特性がお客様に与える影響を評価することが重要です。

一般的なアンチパターン:

- 結果整合性などのトレードオフがある場合でも、すべてのパフォーマンス改善が実装されるべきであると考えている。
- パフォーマンスの問題が限界に達した場合にのみ、ワークロードの変更を評価する。

このベストプラクティスを活用するメリット: パフォーマンス関連の改善の可能性を評価する場合は、変更のトレードオフがワークロード要件で許容できるかどうかを判断する必要があります。場合によっては、トレードオフを補うために追加のコントロールを実装する必要があります。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

パフォーマンスと顧客への影響の観点から、アーキテクチャの重要な領域を特定します。どのように改善できるか、その改善によってどのようなトレードオフがもたらされるか、およびそれらがシステムとユーザーエクスペリエンスにどのように影響するかを判断します。例えば、データのキャッシングを実装すると、パフォーマンスが劇的に向上しますが、システムの不正な動作を防止するためにキャッシュデータをいつどのように更新または無効化するかに関する明確な戦術が必要になります。

実装手順

- ワークロード要件と SLA を理解します。
- 評価要因を明確に定義します。要因は、ワークロードのコスト、信頼性、セキュリティ、パフォーマンスに関連する場合があります。
- 要件に対応できるアーキテクチャとサービスを選択します。
- 実験と概念実証 (POC) を実施して、トレードオフ要因と顧客やアーキテクチャ効率への影響を評価します。通常、可用性とパフォーマンスが高く、安全なワークロードは、顧客体験を向上させるものの、消費するクラウドリソースは多くなります。ワークロードの複雑さ、パフォーマンス、コストのトレードオフを理解します。通常、2 つの要素に優先順位を付けると、3 つ目の要素が犠牲になります。

リソース

関連するドキュメント:

- Amazon Builders' Library
- Amazon QuickSight KPI
- Amazon CloudWatch RUM
- X-Ray ドキュメント
- Understand resiliency patterns and trade-offs to architect efficiently in the cloud

関連動画:

- Optimize applications through Amazon CloudWatch RUM
- AWS re:Invent 2023 - Capacity, availability, cost efficiency: Pick three
- AWS re:Invent 2023 - Advanced integration patterns & trade-offs for loosely coupled systems

関連する例:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP05 ポリシーとリファレンスアーキテクチャを使用する

サービスと設定を選択するときは、ワークロードの設計と実装をより効率的に行うために、社内ポリシーと既存のリファレンスアーキテクチャを使用します。

一般的なアンチパターン:

- 会社の管理諸経費に影響を与える可能性のある幅広い種類のテクノロジーを許可している。

このベストプラクティスを活用するメリット: アーキテクチャ、テクノロジー、ベンダーの選択に関するポリシーを確立することで、迅速な意思決定が可能になります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

リソースとアーキテクチャを選択する際の内部ポリシーを設けることで、アーキテクチャを選択する際に従うべき標準とガイドラインが得られます。これらのガイドラインは、適切なクラウドサービスを選択する際の意思決定プロセスを合理化し、パフォーマンス効率の向上に役立ちます。ポリシーまたはリファレンスアーキテクチャを使用してワークロードをデプロイし、サービスをクラウドデプロイに統合します。次に、パフォーマンステストを使用して、引き続きパフォーマンス要件を満たせることを確認します。

実装手順

- クラウドワークロードの要件を明確に把握します。
- 社内ポリシーと外部ポリシーを確認し、最も関連性の高いポリシーを特定します。
- AWS または業界のベストプラクティスで提供されている適切なリファレンスアーキテクチャを使用します。
- ポリシー、標準、リファレンスアーキテクチャ、および一般的な状況に対応する規範的なガイドラインで構成される一連の流れを作成します。そうすることで、チームがより迅速に行動できるようになります。該当する場合は、業種に合わせてアセットを調整します。

- サンドボックス環境のワークロードに対して、これらのポリシーとリファレンスアーキテクチャを検証します。
- 業界標準や AWS の最新情報を常に把握して、ポリシーとリファレンスアーキテクチャがクラウドワークロードの最適化に役立つことを確認します。

リソース

関連するドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS ナレッジセンター
- AWS Architecture Blog

関連動画:

- This is my Architecture
- AWS re:Invent 2022 - Accelerate value for your business with SAP & AWS reference architecture

関連する例:

- AWS Samples
- AWS SDK サンプル

PERF01-BP06 ベンチマークを使用してアーキテクチャに関する意思決定を行う

既存のワークロードのパフォーマンスをベンチマークに照らして評価すると、クラウドでのパフォーマンスを把握し、そのデータに基づいてアーキテクチャに関する意思決定を行うことができます。

一般的なアンチパターン:

- ワークロードの特性を反映していない一般的なベンチマークを使用している。

- 顧客からのフィードバックと認識を唯一のベンチマークとして使用している。

このベストプラクティスを確立するメリット: 現在の実装をベンチマークに照らして評価することで、パフォーマンスの向上を測定できます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

総合テストでベンチマークを使用して、ワークロードのコンポーネントがどのように機能するかを評価します。ベンチマークは概して負荷テストよりも迅速にセットアップでき、特定のコンポーネントに対するテクノロジーを評価するために使用されます。ベンチマークは、まだ負荷テストができるほどソリューションが完成していないプロジェクトの初期段階によく使用されます。

ワークロードのベンチマーキングには、独自のカスタムベンチマークテストを構築するか、TPC-DS などの業界標準テストを使用できます。業界ベンチマークは、環境を比較する場合に有用です。カスタムベンチマークは、アーキテクチャで採用する予定の特殊なオペレーションを対象にするときに役立ちます。

ベンチマーキングを実施するときは、有効な結果が得られるようにテスト環境の暖気運転を行うことが重要です。同じベンチマークを複数回実行して、時系列での変動を捉えるようにしてください。

ベンチマークは概して負荷テストよりも速く実行されるため、デプロイパイプラインの早い時期に使用でき、パフォーマンスの逸脱に関するフィードバックもより迅速に提供されます。ベンチマークは、コンポーネント、またはサービスにおける大幅な変更を評価する場合に、その変更を行う労力を正当化できるかどうかを見極める近道となり得ます。負荷テストでは、ワークロードが本番環境でどのように機能するかに関する情報が得られることから、ベンチマークは負荷テストと併せて使用することが重要です。

実装手順

- 計画と定義:
  - ベンチマークの目標、ベースライン、テストのシナリオ、メトリクス (CPU 使用率、レイテンシー、スループットなど)、および KPI を定義します。
  - ユーザーエクスペリエンスに関するユーザー要件や、応答時間やアクセシビリティなどの要素に焦点を当てます。
  - ワークロードに適したベンチマークツールを特定します。AWS のサービス (Amazon CloudWatch など) またはワークロードと互換性のあるサードパーティーツールを使用できます。

- 設定とインストルメント化:
  - 環境を設定し、リソースを設定します。
  - モニタリングとログ記録を実装してテスト結果をキャプチャします。
- ベンチマークとモニタリング:
  - ベンチマークテストを実行し、テスト中のメトリクスをモニタリングします。
- 分析と文書化:
  - ベンチマークプロセスと調査結果を文書化します。
  - 結果を分析して、ボトルネック、傾向、改善が必要な領域を特定します。
  - テスト結果を使用して構造に関する意思決定を行い、ワークロードを調整します。これには、サービスの変更や新機能の導入が含まれる場合があります。
- 最適化と反復:
  - ベンチマークに基づいてリソースの設定と割り当てを調整します。
  - 調整後にワークロードを再テストして、改善点を検証します。
  - 学んだことを文書化し、このプロセスを繰り返して改善すべき他の領域を特定します。

リソース

関連するドキュメント:

- AWS アーキテクチャセンター
- AWS Partner Network
- AWS ソリューションライブラリ
- AWS 情報センター
- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- Genomics workflows, Part 5: automated benchmarking
- Benchmark and optimize endpoint deployment in Amazon SageMaker JumpStart

関連動画:

- AWS re:Invent 2023 - Benchmarking AWS Lambda cold starts
- Benchmarking stateful services in the cloud

- This is my Architecture
- Optimize applications through Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics のデモ

関連する例:

- AWS Samples
- AWS SDK のサンプル
- 分散負荷テスト
- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client

PERF01-BP07 データ駆動型のアプローチでアーキテクチャを選択する

アーキテクチャを選択するための明確でデータ駆動型のアプローチを定義して、特定のビジネスニーズを満たす適切なクラウドサービスと設定が使用されていることを確認します。

一般的なアンチパターン:

- 現在のアーキテクチャが静的であり、今後更新されないと考えている。
- 推測や仮定を基にアーキテクチャを選択している。
- あなたは、理由なしで、時間の経過とともにアーキテクチャの変更を導入します。

このベストプラクティスを活用するメリット: アーキテクチャを選択するための明確なアプローチを持つことで、ワークロードの設計にデータを活用し、情報に基づいた意思決定を長期的に行うことができます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

クラウドに関する社内の経験や知識、あるいは公開されているユースケース、関連ドキュメント、ホワイトペーパーなどの外部リソースを利用して、アーキテクチャのリソースとサービスを選択しま

す。ワークロードで利用できるサービスについて、実験とベンチマークを促す明確なプロセスを用意してください。

重要なワークロードのバックログには、ビジネスやユーザーに関連する機能を提供するユーザー関連の背景情報だけでなく、ワークロードのアーキテクチュラルランウェイを形成するテクニカルな背景情報も含める必要があります。このランウェイは、テクノロジーの進歩と新しいサービスからの情報を基に形成され、データと適切な理由に基づいてこうしたテクノロジーやサービスが採用されます。これにより、アーキテクチャの将来性が保たれ、停滞化を防ぐことができます。

実装手順

- 主要な利害関係者と協力して、パフォーマンス、可用性、コストに関する考慮事項を反映したワークロード要件を定義します。ユーザー数やワークロードの使用パターンなどの要素を考慮してください。
- アーキテクチュラルランウェイやテクノロジーバックログを作成して、機能的バックログとともに優先順位を付けます。
- さまざまなクラウドサービスを審査および評価します (詳細については、 PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解するを参照)。
- マイクロサービスやサーバーレスなど、パフォーマンス要件を満たすさまざまなアーキテクチャパターンを検討します (詳細については、 PERF01-BP02 クラウドプロバイダーまたは適切なパートナーからのガイダンスを使用して、アーキテクチャパターンとベストプラクティスについて学ぶを参照)。

- アーキテクチャ図を参照する、または他のチームやAWS ソリューションアーキテクト、AWS アーキテクチャセンターなどのリソースに問い合わせてください。また、 AWS Partner Networkは、ワークロードに適したアーキテクチャを選択するのに役立ちます。

- ワークロードのパフォーマンスを評価するのに役立つスループットや応答時間などのパフォーマンスメトリクスを定義します。
- 定義したメトリクスを試用し、選択したアーキテクチャのパフォーマンスを検証します。
- アーキテクチャの最適なパフォーマンスを維持するために、継続的にモニタリングし、必要に応じて調整を行います。
- 選択したアーキテクチャと決定事項を、今後のアップデートや学習の参考として文書化します。
- 学習したことや新しいテクノロジー、さらに現在のアプローチで必要とされる変更や問題を示す指標に基づいて、アーキテクチャの選択アプローチを継続的に見直し、更新します。

リソース

関連するドキュメント:

- AWS ソリューションライブラリ
- AWS ナレッジセンター
- Architectural Patterns to Build End-to-End Data Driven Applications on AWS

関連動画:

- This is my Architecture
- AWS re:Invent 2021 - Data-driven enterprise: Going from vision to value
- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures
- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results

関連する例:

- AWS Samples
- AWS SDK サンプル

コンピューティングとハードウェア

特定のワークロードに対する最適なコンピューティングの選択は、アプリケーションの設計、利用パターン、および構成設定に応じて異なります。アーキテクチャでは、各種コンポーネントに異なるコンピューティングを使用し、異なる機能を有効化してパフォーマンスを向上させることができます。アーキテクチャに誤ったコンピューティングを選択することは、パフォーマンス効率の低下につながる可能性があります。

この重点分野では、クラウドでのパフォーマンス効率を高めるために、コンピューティングオプションを特定して最適化する方法に関するガイダンスとベストプラクティスを紹介します。

ベストプラクティス

- PERF02-BP01 ワークロードに最適なコンピューティングオプションを選択する
- PERF02-BP02 利用可能なコンピューティング設定と機能について理解する
- PERF02-BP03 コンピューティング関連のメトリクスを収集する
- PERF02-BP04 コンピューティングリソースの設定とライトサイジングを行う
- PERF02-BP05 コンピューティングリソースを動的にスケールする
- PERF02-BP06 最適化されたハードウェアベースのコンピューティングアクセラレーターを使用する

PERF02-BP01 ワークロードに最適なコンピューティングオプションを選択する

ワークロードに最適なコンピューティングオプションを選択することで、パフォーマンスを高め、不要なインフラストラクチャコストを削減し、ワークロードを維持するために必要な運用工数を軽減できます。

一般的なアンチパターン:

- オンプレミスで使用されていたものと同じコンピューティングオプションを使用している。
- クラウドコンピューティングのオプション、機能、ソリューション、およびそうしたソリューションがコンピューティング性能の向上にどのように役立つかについての認識が足りない。
- ワークロードの特性により的確に適合する代替のコンピューティングオプションがあるにもかかわらず、スケーリングやパフォーマンスの要件を満たすために既存のコンピューティングオプションを過剰にプロビジョニングしている。

このベストプラクティスを確立するメリット: コンピューティング要件を特定し、利用可能なオプションに照らし合わせて評価することで、ワークロードのリソース効率を高めることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

クラウドワークロードを最適化してパフォーマンスを効率化するには、ユースケースとパフォーマンス要件に最適なコンピューティングオプションを選択することが重要です。AWS では、クラウド内のさまざまなワークロードに対応するさまざまなコンピューティングオプションを用意しています。例えば、Amazon EC2 を使用して仮想サーバーを起動して管理する、AWS Lambda を使用してサーバーのプロビジョニングや管理を行うことなくコードを実行する、Amazon ECS や Amazon EKS を使用してコンテナを実行して管理する、AWS Batch を使用して大量のデータを並列処理するといったことが可能です。スケールとコンピューティングニーズを基に、状況に最適なコンピューティングソリューションを選んで構成する必要があります。また、コンピューティングソリューションにもそれぞれ利点と欠点があるため、1 つのワークロードで複数のタイプを使用することも検討できます。

次の手順では、ワークロードの特性とパフォーマンス要件に合わせて適切なコンピューティングオプションを選択する方法を説明します。

実装手順

- ワークロードのコンピューティング要件を把握します。主な要件には、処理ニーズ、トラフィックパターン、データアクセスパターン、スケーリングの必要性、レイテンシー要件があります。
- AWS 上のワークロードで使用できるさまざまなコンピューティングオプションについて学びます (概要については、「PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する」を参照してください)。AWS の主要なコンピューティングオプション、その特徴、一般的な使用例は次のとおりです。

AWS service	Key characteristics	Common use cases
Amazon Elastic Compute Cloud (Amazon EC2)	Has dedicated option for hardware, license requirements, large selection of different instance families, processor types and compute accelerators	Lift and shift migrations, monolithic application, hybrid environments, enterprise applications

AWS service	Key characteristics	Common use cases
Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS)	Easy deployment, consistent environments, scalable	Microservices, hybrid environments
AWS Lambda	サーバーレスコンピューティング service that runs code in response to events and automatically manages the underlying compute resources.	Microservices, event-driven applications
AWS Batch	Efficiently and dynamically provisions and scales Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), and AWS Fargate compute resources, with an option to use On-Demand or Spot Instances based on your job requirements	HPC, train ML models
Amazon Lightsail	Preconfigured Linux and Windows application for running small workloads	Simple web applications, custom website

- 各コンピューティングオプションに関連するコスト (時間単位の料金やデータ転送など) と管理諸経費 (パッチ適用やスケーリングなど) を評価します。
- 非運用環境で実験とベンチマーキングを行い、どのコンピューティングオプションがワークロード要件に最も適しているかを特定します。
- 実験を通じて新しいコンピューティングソリューションを特定したら、移行を計画し、パフォーマンスメトリクスを検証します。

- AWS が提供する Amazon CloudWatch などのモニタリングツールや AWS Compute Optimizer などの最適化サービスを使用して、実際の使用パターンに基づいてコンピューティングリソースを継続的に最適化します。

リソース

関連するドキュメント:

- AWS を使用したクラウドコンピューティング
- Amazon EC2 インスタンスタイプ
- Amazon EKS コンテナ: Amazon EKS ワーカーノード
- Amazon ECS コンテナ: Amazon ECS コンテナインスタンス
- 関数: Lambda 関数の設定
- コンテナに関する規範ガイダンス
- サーバーレスに関する規範ガイダンス

関連動画:

- AWS re:Invent 2023 - AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 - New Amazon Elastic Compute Cloud generative AI capabilities in AMS
- AWS re:Invent 2023 - What's new with Amazon Elastic Compute Cloud
- AWS re:Invent 2023 - Smart savings: Amazon Elastic Compute Cloud cost-optimization strategies
- AWS re:Invent 2021 - Powering next-gen Amazon Elastic Compute Cloud: Deep dive on the Nitro System
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute
- AWS re:Invent 2019 - Amazon Elastic Compute Cloud foundations
- AWS re:Invent 2022 - Deploy ML models for inference at high performance and low cost
- AWS re:Invent 2019 - Optimize performance and cost for your AWS compute
- Amazon EC2 foundations
- Deploy ML models for inference at high performance and low cost

関連する例:

- Migrating the Web application to containers
- Run a Serverless Hello World
- Amazon EKS Workshop
- Amazon EC2 Workshop
- Efficient and Resilient Workloads with Amazon Elastic Compute Cloud Auto Scaling
- Migrating to AWS Graviton with Container Services

PERF02-BP02 利用可能なコンピューティング設定と機能について理解する

コンピューティングサービスで利用できる設定オプションと機能を理解しておくと、適切な量のリソースをプロビジョニングしてパフォーマンス効率を向上させることができます。

一般的なアンチパターン:

- コンピューティングオプションや利用可能なインスタンスファミリーをワークロードの特性に照らして評価しない。
- ピーク需要の要件を満たすために、コンピューティングリソースを過剰にプロビジョニングしている。

このベストプラクティスを活用するメリット: AWS のコンピューティングの機能と設定に精通していれば、ワークロードの特性とニーズに合わせて最適化されたコンピューティングソリューションを使用できます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

各コンピューティングソリューションは、さまざまなワークロードの特性と要件に対応するために、それぞれ異なる設定や機能を備えています。こうしたオプションがワークロードをどのように補完するかを理解し、アプリケーションにどの設定オプションが最適か判断します。これらのオプションの例には、インスタンスのファミリー、サイズ、機能 (GPU、I/O)、バースト、タイムアウト、関数サイズ、コンテナインスタンス、並行性などがあります。ワークロードで同じコンピューティングオプションを 4 週間以上使用しており、その特性が今後も変わらないと予想される場合は、 AWS Compute Optimizer を使用して、現在のコンピューティングオプションがワークロードに適しているかどうかを、CPU とメモリの観点から確認できます。

実装手順

1. ワークロード要件 (CPU ニーズ、メモリ、レイテンシーなど) を把握します。
2. AWS ドキュメントとベストプラクティスで、コンピューティングパフォーマンスの向上に役立つ推奨構成オプションについて確認します。考慮すべき主な設定オプションは次のとおりです。

設定オプション	例
インスタンスタイプ	• コンピューティング最適化 インスタンスは、高い vCPU 対メモリ比を必要とするワークロードに最適です。 • メモリ最適化 インスタンスは大量のメモリを提供し、メモリを集中的に使用するワークロードをサポートします。 • ストレージ最適化 インスタンスは、ローカルストレージへの高いシーケンシャルな読み書きアクセス (IOPS) を必要とするワークロード向けに設計されています。
料金モデル	• オンデマンドインスタンス では、1 時間または 1 秒単位でコンピューティング性能を利用でき、長期的な契約は必要ありません。このインスタンスは、パフォーマンスのベースラインを超えるようなバースト的なニーズに適しています。 • Savings Plans を使用すると、1 年または 3 年の単位で一定量のコンピューティング容量を確約することで、オンデマンドインスタンスと比較して大幅にコストを削減できます。 • スポットインスタンス では、ステートレスで耐障害性のあるワークロードで、未使用のインスタンス容量を割引価格で利用できます。

設定オプション	例
Auto Scaling	Auto Scaling 設定 を使用して、 コンピューティングリソースをトラフィックパターンに一致させます。
サイジング	• Compute Optimizer を使用すると、 機械学習によって、コンピューティング特性に最適なコンピューティング設定についての推奨事項が提示されます。 • AWS Lambda Power Tuning を使用して、 Lambda 関数に最適な設定を選択します。
ハードウェアベースのコンピューティングアクセラレーター	• 高速コンピューティングインスタンスでは、 CPU ベースの代替インスタンスよりも効率的にグラフィック処理やデータパターンマッチングなどの機能を実行できます。 • 機械学習のワークロードには、 AWS Trainium、 AWS Inferentia、 Amazon EC2 DL1 など、ワークロードに特化した専用ハードウェアを利用します

リソース

関連ドキュメント:

- AWS を使用したクラウドコンピューティング
- Amazon EC2 インスタンスタイプ
- Amazon EC2 インスタンスのプロセッサのステート制御
- Amazon EKS コンテナ: Amazon EKS ワーカーノード
- Amazon ECS コンテナ: Amazon ECS コンテナインスタンス
- Functions: Lambda Function Configuration (関数: Lambda 関数の設定)

関連動画:

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations
- AWS re:Invent 2022 – https://www.youtube.com/watch?v=5B4-s_ivn1o

関連する例:

- Compute Optimizer デモコード
- Amazon EC2 スポットインスタンスワークショップ
- Efficient and Resilient Workloads with Amazon EC2 AWS Auto Scaling
- Graviton デベロッパーワークショップ
- AWS for Microsoft workloads immersion day
- AWS for Linux workloads immersion day
- AWS Compute Optimizer デモコード
- Amazon EKS Workshop

PERF02-BP03 コンピューティング関連のメトリクスを収集する

コンピューティング関連のメトリクスを記録および追跡することで、コンピューティングリソースのパフォーマンスをよりよく理解し、パフォーマンスと使用率を向上させます。

一般的なアンチパターン:

- メトリクスの検索に手動ログファイルのみを使用している。
- 一部のモニタリングソフトウェアで記録されるデフォルトのメトリクスのみを使用している。
- 問題が発生したときにだけメトリクスを確認している。

このベストプラクティスを活用するメリット: パフォーマンス関連のメトリクスを収集することで、アプリケーションのパフォーマンスとビジネス要件の整合性をとり、ワークロードのニーズを満たす

ことができます。また、ワークロードにおけるリソースのパフォーマンスと使用率を継続的に改善するのにも役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

クラウドワークロードでは、メトリクス、ログ、イベントなどのデータが大量に生成される可能性があります。AWS クラウド では、メトリクスの収集は、セキュリティ、コスト効率、パフォーマンス、持続可能性を向上させるために不可欠なステップです。AWS では、 Amazon CloudWatch のようなモニタリングサービスを使用して、有益なインサイトにつながるさまざまなパフォーマンス関連のメトリクスを提供しています。CPU 使用率、メモリ使用率、ディスク I/O、ネットワークのインバウンドとアウトバウンドなどのメトリクスにより、使用率レベルやパフォーマンスのボトルネックを把握できます。これらのメトリクスをデータ駆動型のアプローチの一部として使用し、ワークロードのリソースを積極的に調整および最適化します。 理想は、コンピューティングリソースに関連するすべてのメトリクスを単一のプラットフォームで収集し、コストと運用上の目標をサポートするための保持ポリシーを実装することです。

実装手順

1. ワークロードに関連するパフォーマンス関連の指標を特定します。リソース使用率やクラウドワークロードの動作状況 (応答時間やスループットなど) に関するメトリクスを収集する必要があります。
   a. Amazon EC2 のデフォルトのメトリクス
   b. Amazon ECS のデフォルトのメトリクス
   c. Amazon EKS のデフォルトのメトリクス
   d. Lambda のデフォルトのメトリクス
   e. Amazon EC2 のメモリとディスクのメトリクス
2. ワークロードに適したログ記録とモニタリングのソリューションを選んでセットアップします。
   a. AWS native Observability
   b. AWS Distro for OpenTelemetry
   c. Amazon Managed Service for Prometheus
3. ワークロード要件に基づいて、メトリクスに必要なフィルターと集計を定義します。
   a. Quantify custom application metrics with Amazon CloudWatch Logs and metric filters
   b. Collect custom metrics with Amazon CloudWatch strategic tagging

4. セキュリティと運用の目標に合わせて、メトリクスのデータ保持ポリシーを設定します。
    a. CloudWatch メトリクスのデフォルトのデータ保持
    b. CloudWatch Logs のデフォルトのデータ保持
5. パフォーマンス関連の問題に積極的に対応できるよう、必要に応じて、メトリクスのアラームと通知を作成します。
    a. Create alarms for custom metrics using Amazon CloudWatch anomaly detection
    b. Create metrics and alarms for specific web pages with Amazon CloudWatch RUM
6. 自動化を利用して、メトリクス・ログ集計エージェントをデプロイします。
    a. AWS Systems Manager automation
    b. OpenTelemetry Collector

リソース

関連するドキュメント:

- モニタリングとオブザーバビリティ
- Best practices: implementing observability with AWS
- Amazon CloudWatch のドキュメント
- CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからのメトリクスとログを収集する
- Accessing Amazon CloudWatch Logs for AWS Lambda
- コンテナインスタンスでの CloudWatch Logs の使用
- カスタムメトリクスをパブリッシュする
- AWS の回答: 集中ログ記録
- CloudWatch メトリクスを発行する AWS のサービス
- AWS Fargate での Amazon EKS のモニタリング

関連動画:

- AWS re:Invent 2023 – [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 – Implementing application observability
- AWS re:Invent 2023 – Building an effective observability strategy

- AWS re:Invent 2023 – Seamless observability with AWS Distro for OpenTelemetry
- Application Performance Management on AWS

関連する例:

- AWS for Linux Workloads Immersion Day- Amazon CloudWatch
- Monitoring Amazon ECS clusters and containers
- Monitoring with Amazon CloudWatch dashboards
- Amazon EKS Workshop

PERF02-BP04 コンピューティングリソースの設定とライトサイジングを行う

ワークロードのパフォーマンス要件に合わせてコンピューティングリソースの設定とライトサイジングを行うことで、リソースの過不足を防ぎます。

一般的なアンチパターン:

- ワークロードのパフォーマンス要件を無視した結果、コンピューティングリソースのプロビジョニングが過剰になったり不足したりする。
- 使用できる最大または最小のインスタンスのみをすべてのワークロードに対して選択する。
- 管理を容易にするため、1 つのインスタンスファミリーのみを使用する。
- AWS Cost Explorer または Compute Optimizer からのライトサイジングに関する推奨事項を無視する。
- 新しいインスタンスタイプが適合するかどうかについてワークロードを再評価しない。
- 組織で使用できるインスタンス設定としてごく少数のみを認証する。

このベストプラクティスを活用するメリット: コンピューティングリソースをライトサイジングすることで、リソースの過剰プロビジョニングやプロビジョニング不足を回避できるため、クラウドでの運用が最適化されます。通常、コンピューティングリソースのサイズを適切に設定すると、パフォーマンスが上がり、カスタマーエクスペリエンスが向上すると同時に、コストも削減されます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

ライトサイジングにより、組織はビジネスニーズに対応しながら、効率的かつ費用対効果の高い方法でクラウドインフラストラクチャを運用できます。クラウドリソースを過剰にプロビジョニングすると余分なコストが発生する可能性があり、プロビジョニングが不十分だと、パフォーマンスが低下し、カスタマーエクスペリエンスが低下する可能性があります。AWS の AWS Compute Optimizer および AWS Trusted Advisor のようなツールは、履歴データを使用して、コンピューティングリソースを適切なサイズにするための推奨事項を提供します。

実装手順

- ニーズに最適なインスタンスタイプを選択します。
  - ワークロードに適切な Amazon EC2 インスタンスタイプを選択する方法を教えてください。
  - Amazon EC2 フリートの属性ベースのインスタンスタイプの選択
  - 属性ベースのインスタンスタイプの選択を使用して Auto Scaling グループを作成する
  - Optimizing your Kubernetes compute costs with Karpenter consolidation
- ワークロードのさまざまなパフォーマンス特性と、それらの特性とメモリ、ネットワーク、CPU 使用率との関連を分析します。このデータを使用して、ワークロードのプロファイルとパフォーマンス目標に最適なリソースを選択します。
- Amazon CloudWatch などのモニタリングツールを使用して、AWS リソースの使用状況をモニタリングします。
- コンピューティングリソースの適切な構成を選択します。
  - 一時的なワークロードについては、インスタンス Amazon CloudWatch メトリクス (`CPUUtilization` など) を評価して、インスタンスの過剰または過少な使用を特定します。
  - 安定したワークロードの場合は、AWS のライトサイジングツール (AWS Compute Optimizer、AWS Trusted Advisor など) を定期的にチェックし、インスタンスの最適化とライトサイジングの機会を特定します。
- 構成の変更は、本番環境に実装する前に非運用環境でテストします。
- 継続的に新しいコンピューティングサービスを再評価し、ワークロードのニーズと照らし合わせます。

リソース

関連ドキュメント:

- AWS を使用したクラウドコンピューティング
- Amazon EC2 インスタンスタイプ
- Amazon ECS コンテナ: Amazon ECS コンテナインスタンス
- Amazon EKS コンテナ: Amazon EKS ワーカーノード
- 関数: Lambda 関数の設定
- Amazon EC2 インスタンスのプロセッサのステート制御

関連動画:

- Amazon EC2 foundations
- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

関連する例:

- AWS Compute Optimizer デモコード
- Amazon EKS Workshop
- Right-sizing recommendations

PERF02-BP05 コンピューティングリソースを動的にスケールする

クラウドの伸縮性を利用して、ニーズに合わせてコンピューティングリソースを動的にスケールアップまたはスケールダウンすることで、ワークロードのキャパシティが過剰または過少になるのを防ぐことができます。

一般的なアンチパターン:

- アラームに対応するために手動でキャパシティを増やす。
- オンプレミスと同じサイズ設定ガイドライン (通常は静的インフラストラクチャ) を使用する。

- スケーリングイベントの後、スケールダウンして元に戻すのではなく、キャパシティを増加させたままにする。

このベストプラクティスを活用するメリット: コンピューティングリソースの伸縮性を設定してテストすることで、コストの節約、パフォーマンスベンチマークの維持、トラフィックの変化に応じた信頼性の向上に役立ちます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

AWS は、需要の変化に対応するためのさまざまなスケーリングメカニズムを通じて、リソースを動的にスケールアップまたはスケールダウンする柔軟性を備えています。コンピューティング関連のメトリクスと組み合わせると、動的スケーリングにより、ワークロードが自動的に変化に対応し、最適なコンピューティングリソースを使用して目標を達成できるようになります。

リソースの需要と供給は、さまざまなアプローチで一致させることができます。

- ターゲット追跡アプローチ: スケーリングメトリクスをモニタリングし、必要に応じて容量を自動的に増減します。
- 予測スケーリング: 日単位および週単位の傾向を見越してスケールします。
- スケジュールベースのアプローチ: 予測できる負荷の変化に従って、独自のスケーリングスケジュールを設定します。
- サービススケーリング: 設計により自動的にスケーリングされるサービス (サーバーレスなど) を選択します。

ワークロードのデプロイメントで、確実にスケールアップおよびスケールダウンイベントを対処できるようにしてください。

実装手順

- コンピューティングインスタンス、コンテナ、関数のいずれにも、伸縮性の仕組みが備わっています。サービスの機能として実装されている場合も、自動スケーリングと組み合わせて実現する場合もあります。自動スケーリングメカニズムの例を次に示します。

自動スケーリングメカニズム	使用する場所
Amazon EC2 Auto Scaling	アプリケーションのユーザー負荷の処理に必要な適切な数の Amazon EC2 インスタンスを確保する。
Application Auto Scaling	Amazon Elastic Container Service 関数や Amazon Elastic Container Service (Amazon ECS) サービスなど、Amazon ECS 以外の個別の Amazon Elastic Container Service サービス を 自動的に スケーリング する。
Kubernetes Cluster Autoscaler/Karpenter	Kubernetes クラスターを自動的にスケーリングする。

- スケーリングは大抵、AWS Glue インスタンスや AWS Lambda 関数などのコンピューティングサービスに関連して取り上げられます。AWS Glue のようなコンピューティング以外のサービスの設定も、 需要に合わせて 考慮するようにしてください。
- スケーリングのメトリクスが、デプロイされているワークロードの特性と一致していることを確認します。動画トランスコーディングアプリケーションをデプロイしようとする場合、100% の CPU 使用率が想定されるため、プライマリメトリクスにするべきではありません。代わりに、トランスコーディングジョブのキュー深度を使用してください。必要な場合は、スケーリングポリシーとして カスタマイズされたメトリクス を使用できます。適切なメトリクスを選ぶには、Amazon EC2 の以下のガイダンスを考慮してください。
  - メトリクスは有効な利用率メトリクスでなければならず、インスタンスのどの程度ビジーかを記述する必要があります。
  - メトリクス値は、Auto Scaling グループ内のインスタンス数に比例して増減する必要があります。
- 必ず 手動スケーリング ではなく 動的スケーリング を Auto Scaling グループに使用します。また、動的スケーリングでは、 ターゲット追跡スケーリングポリシー を使用することをお勧めします。
- ワークロードのデプロイがスケーリングイベント (アップとダウン) の両方に対応処理できることを確認します。例えば、 アクティビティ履歴 を使用して、Auto Scaling グループのスケーリングアクティビティを確認することができます。

- ワークロードを評価して予測可能なパターンを見つけ、あらかじめわかっていた、および計画的な需要の変化を予測してプロアクティブにスケールします。予測スケーリングを使用すると、容量を過剰にプロビジョニングする必要がなくなります。詳細については、「Amazon EC2 Auto Scaling での予測スケーリング」を参照してください。

リソース

関連ドキュメント:

- AWS を使用したクラウドコンピューティング
- Amazon EC2 インスタンスタイプ
- Amazon ECS コンテナ: Amazon ECS コンテナインスタンス
- Amazon EKS コンテナ: Amazon EKS ワーカーノード
- 関数: Lambda 関数の設定
- Amazon EC2 インスタンスのプロセッサのステート制御
- Amazon ECS クラスター Auto Scaling の詳細
- Introducing Karpenter – An Open-Source High-Performance Kubernetes Cluster Autoscaler

関連動画:

- AWS re:Invent 2023 – AWS Graviton: The best price performance for your AWS workloads
- AWS re:Invent 2023 – New Amazon EC2 generative AI capabilities in AWS Management Console
- AWS re:Invent 2023 – What's new with Amazon EC2
- AWS re:Invent 2023 – Smart savings: Amazon EC2 cost-optimization strategies
- AWS re:Invent 2021 – Powering next-gen Amazon EC2: Deep dive on the Nitro System
- AWS re:Invent 2019 – Amazon EC2 foundations

関連する例:

- Amazon EC2 Auto Scaling グループの例
- Amazon EKS Workshop
- Scale your Amazon EKS workloads by running on IPv6

PERF02-BP06 最適化されたハードウェアベースのコンピューティングアクセラレーターを使用する

ハードウェアアクセラレーターを使用すると、CPU ベースの代替手段よりも効率的に特定の機能を実行できます。

一般的なアンチパターン:

- ワークロードで、より高いパフォーマンスとより低いコストを実現できる専用のインスタンスに対する汎用インスタンスのベンチマーキングを行っていない。
- CPU ベースのコンピューティングアクセラレーターを使用した方が効率的なタスクに、ハードウェアベースのコンピューティングアクセラレーターを使用している。
- GPU の使用状況を監視していない。

このベストプラクティスを確立するメリット: GPU (グラフィックス処理ユニット) や FPGA (フィールドプログラマブルゲートアレイ) などのハードウェアベースのアクセラレーターを使用することで、特定の処理機能をより効率的に実行できます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

高速コンピューティングインスタンスを使用すると、GPU や FPGA などのハードウェアベースのコンピューティングアクセラレーターにアクセスできます。これらのハードウェアアクセラレーターは、グラフィック処理やデータパターンマッチングなどの特定の機能を、CPU ベースの代替手段よりも効率的に実行します。レンダリング、トランスコーディング、機械学習など、多くの高速ワークロードは、リソースの使用量に大きなばらつきがあります。このハードウェアは必要な時間だけ実行し、必要のない場合は自動で廃止することで、全体的なパフォーマンス効率を向上させることができます。

実装手順

- どの高速コンピューティングインスタンスが要件に対応できるかを特定します。
- 機械学習のワークロードには、AWS Trainium、AWS Inferentia、Amazon EC2 DL1 など、ワークロードに特化した専用ハードウェアを利用します。Inf2 インスタンスなどの AWS Inferentia インスタンスは、同等の Amazon EC2 インスタンスと比較してワットあたりのパフォーマンスが最大 50% 向上します。

- 高速コンピューティングインスタンスの使用状況メトリクスを収集します。例えば、Amazon CloudWatch で NVIDIA GPU メトリクスを収集するに示すように、CloudWatch エージェントを使用して GPU の `utilization_gpu` や `utilization_memory` などのメトリクスを収集できます。
- ハードウェアアクセラレーターのコード、ネットワーク操作、設定を最適化し、基盤となるハードウェアが十分に活用されるようにします。
  - GPU 設定を最適化する
  - Deep Learning AMI での GPU のモニタリングと最適化
  - Amazon SageMaker における深層学習トレーニングの GPU パフォーマンスチューニングのための I/O の最適化
- 最新の高性能ライブラリと GPU ドライバーを使用します。
- 使用しないときは、自動化を使用して GPU インスタンスを解放します。

リソース

関連するドキュメント:

- Working with GPUs on Amazon Elastic Container Service
- 分散された機械学習トレーニング
- AWS Trainium を搭載したインスタンス
- AWS Inferentia を搭載したインスタンス
- Let's Architect! Architecting with custom chips and accelerators

- 高速コンピューティング
- Amazon EC2 VT1 インスタンス
- ワークロードに適切な Amazon EC2 インスタンスタイプを選択する方法を教えてください。
- Choose the best AI accelerator and model compilation for computer vision inference with Amazon SageMaker

関連動画:

- AWS re:Invent 2021 - How to select Amazon Elastic Compute Cloud GPU instances for deep learning

- AWS re:Invent 2022 - [NEW LAUNCH!] Introducing AWS Inferentia2-based Amazon EC2 Inf2 instances
- AWS re:Invent 2022 - Accelerate deep learning and innovate faster with AWS Trainium
- AWS re:Invent 2022 - Deep learning on AWS with NVIDIA: From training to deployment

関連する例:

- Amazon SageMaker and NVIDIA GPU Cloud (NGC)
- Use SageMaker with Trainium and Inferentia for optimized deep learning training and inferencing workloads
- Optimizing NLP models with Amazon Elastic Compute Cloud Inf1 instances in Amazon SageMaker

データ管理

特定のシステムに最適なデータ管理ソリューションは、データの種類 (ブロック、ファイル、またはオブジェクト)、アクセスパターン (ランダムまたはシーケンシャル)、必要なスループット、アクセス頻度 (オンライン、オフライン、アーカイブ)、更新頻度 (WORM、動的)、および可用性と耐久性に関する制約に応じて異なります。優れた設計のワークロードは、さまざまな機能によってパフォーマンスを向上させることができる専用のデータストアを使用します。

この重点分野では、データの保管、移動とアクセスのパターン、データストアのパフォーマンス効率を最適化するためのガイダンスとベストプラクティスを紹介します。

ベストプラクティス

- PERF03-BP01 データアクセスとストレージ要件に最適な専用データストアを使用する
- PERF03-BP02 データストアで利用可能な設定オプションを評価する
- PERF03-BP03 データストアのパフォーマンスメトリクスを収集・記録する
- PERF03-BP04 データストアのクエリパフォーマンスを向上させるための戦略を実装する
- PERF03-BP05 キャッシュを利用するデータアクセスパターンを実装する

PERF03-BP01 データアクセスとストレージ要件に最適な専用データストアを使用する

データの特性 (共有可能、サイズ、キャッシュサイズ、アクセスパターン、レイテンシー、スループット、データの持続性など) を理解して、ワークロードに適した専用データストア (ストレージまたはデータベース) を選択します。

一般的なアンチパターン:

- 特定のタイプのデータストアに関する社内知識と経験があるため、1 つのデータベースソリューションに固執する。
- すべてのワークロードのデータの保存とアクセスの要件が類似していると考えている。
- データアセットのインベントリにデータカタログを実装していない。

このベストプラクティスを確立するメリット: データの特性と要件を理解することで、ワークロードのニーズに適した、最も効率的でパフォーマンスの高いストレージテクノロジーを判別できます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

データストレージを選択して実装する際は、クエリ、スケーリング、ストレージの特性がワークロードのデータ要件をサポートしていることを確認します。AWS では、ブロックストレージ、オブジェクトストレージ、ストリーミングストレージ、ファイルシステム、リレーショナル、key-value、ドキュメント、インメモリ、グラフ、時系列、台帳などのデータベースをはじめとした、さまざまなデータストレージとデータベーステクノロジーを提供しています。各データ管理ソリューションには、ユースケースとデータモデルをサポートするために使用できるオプションと設定があります。データの特性と要件を理解することで、モノリシックなストレージテクノロジーや制約の多い汎用的なアプローチから脱却し、データの適切な管理に集中できます。

実装手順

- ワークロードに存在するさまざまなデータタイプを棚卸しします。
- 次のようなデータの特性と要件を理解して文書化します。
  - データタイプ (非構造化、半構造化、リレーショナル)
  - データ量と増加
  - データ保存期間: 永続、一時的、一過性
  - ACID 特性 (原子性、一貫性、独立性、耐久性) の要件
  - データアクセスパターン (読み取りが多い、または書き込みが多い)
  - レイテンシー
  - スループット
  - IOPS (1 秒あたりの入出力操作数)
  - データの保管期間
- AWS 上のワークロードで使用できる各種のデータストアから使用するデータの特性に合致するものを確認します (「PERF01-BP01 利用可能なクラウドサービスと機能について学び、理解する」で説明) 。AWS のストレージ技術とその主な特徴を例としていくつか挙げます。

タイプ	AWS のサービス	主な特徴
Object storage	Amazon S3	Unlimited scalability, high availability, and multiple options for accessibility.

タイプ	AWS のサービス	主な特徴
		Transferring and accessing objects in and out of Amazon S3 can use a service, such as Transfer Acceleration or Access Points, to support your location, security needs, and access patterns.
Archiving storage	Amazon S3 Glacier	Built for data archiving.
Streaming storage	Amazon Kinesis Amazon Managed Streaming for Apache Kafka (Amazon MSK)	Efficient ingestion and storage of streaming data.
Shared file system	Amazon Elastic File System (Amazon EFS)	複数のタイプのコンピューティングソリューションからアクセスできるマウント可能なファイルシステムです。
Shared file system	Amazon FSx	Built on the latest AWS compute solutions to support four commonly used file systems: NetApp ONTAP, OpenZFS, Windows File Server, and Lustre. Amazon FSx レイテンシー、スループット、および IOPS は vary per file system and should be considered when selecting the right file system for your workload needs.

タイプ	AWS のサービス	主な特徴
Block storage	Amazon Elastic Block Store (Amazon EBS)	Scalable, high-performance block-storage service designed for Amazon Elastic Compute Cloud (Amazon EC2). Amazon EBS includes SSD-backed storage for transactional, IOPS-intensive workloads and HDD-backed storage for throughput-intensive workloads.
Relational database	Amazon Aurora, Amazon RDS, Amazon Redshift.	Designed to support ACID (atomicity, consistency, isolation, durability) transactions, and maintain referential integrity and strong data consistency. Many traditional applications, enterprise resource planning (ERP), customer relationship management (CRM), and ecommerce use relational databases to store their data.
Key-value database	Amazon DynamoDB	Optimized for common access patterns, typically to store and retrieve large volumes of data. High-traffic web apps, ecommerce systems, and gaming applications are typical use-cases for key-value databases.

タイプ	AWS のサービス	主な特徴
Document database	Amazon DocumentDB	Designed to store semi-structured data as JSON-like documents. These databases help developers build and update applications such as content management, catalogs, and user profiles quickly.
In-memory database	Amazon ElastiCache , Amazon MemoryDB for Redis	Used for applications that require real-time access to data, lowest latency and highest throughput. You may use in-memory databases for application caching, session management, gaming leaderboards, low latency ML feature store, microservices messaging system, and a high-throughput streaming mechanism
Graph database	Amazon Neptune	Used for applications that must navigate and query millions of relationships between highly connected graph datasets with millisecond latency at large scale. Many companies use graph databases for fraud detection, social networking, and recommendation engines.

タイプ	AWS のサービス	主な特徴
Time Series database	Amazon Timestream	Used to efficiently collect, synthesize, and derive insights from data that changes over time. IoT applications, DevOps, and industrial telemetry can utilize time-series databases.
Wide column	Amazon Keyspaces (Apache Cassandra 向け)	Uses tables, rows, and columns, but unlike a relational database, the names and format of the columns can vary from row to row in the same table. You typically see a wide column store in high scale industrial apps for equipment maintenance, fleet management, and route optimization.
Ledger	Amazon Quantum Ledger Database (Amazon QLDB)	Provides a centralized and trusted authority to maintain a scalable, immutable, and cryptographically verifiabl e record of transactions for every application. We see ledger databases used for systems of record, supply chain, registrations, and even banking transactions.

- データプラットフォームを構築する場合は、AWS の最新のデータアーキテクチャを活用して、データレイク、データウェアハウス、専用データストアを統合します。
- ワークロードのデータストアを選択する際に考慮すべき主なポイントは次のとおりです。

Question	Things to consider
How is the data structured?	• データが構造化されていない場合は、Amazon S3 などのオブジェクトストアまたは Amazon DocumentDB のような NoSQL データベースを検討します。 • key-value データの場合は、DynamoDB、Amazon ElastiCache for Redis、Amazon MemoryDB for Redis を検討します。
What level of referential integrity is required?	• 外部キー制約の場合、Amazon RDS や Aurora などのリレーショナルデータベースがこのレベルの整合性を提供できます。 • 通常、NoSQL データモデル内では、データをドキュメントまたはテーブルをまたいで結合するのではなく、単一のドキュメントまたはドキュメントのコレクションに非正規化して、単一のリクエストで取得します。
Is ACID (atomicity, consistency, isolation, durability) compliance required?	• リレーショナルデータベースに関連付けられている ACID 特性が必要な場合は、Amazon RDS や Aurora を検討します。 • NoSQL データベースに強整合性が必要な場合は、DynamoDB の強力な整合性のある読み込みを使用できます。

Question	Things to consider
How will the storage requirements change over time? How does this impact scalability?	• DynamoDB や Amazon Quantum Ledger Database (Amazon QLDB) などのサーバーレスデータベースは動的にスケールします。 • リレーショナルデータベースには、プロビジョニングされたストレージに上限があり、多くの場合、この上限に達すると、シャーディングなどのメカニズムを使用して水平方向に分割する必要があります。
What is the proportion of read queries in relation to write queries? Would caching be likely to improve performance?	• 読み込みの多いワークロードには、ElastiCache またはデータベースが DynamoDB の場合は DAX などのキャッシュレイヤーが有効です。 • 読み込みは、Amazon RDS のようなリレーショナルデータベースの読み込みレプリカにオフロードすることもできます。
Does storage and modification (OLTP - Online Transaction Processing) or retrieval and reporting (OLAP - Online Analytical Processing) have a higher priority?	• 高スループットのそのまま読み取るトランザクション処理については、DynamoDB などの NoSQL データベースを検討します。 • 一貫性のある高スループットで複雑な読み取りパターン (join など) には Amazon RDS を使用します。 • 分析クエリの場合は、Amazon Redshift などの列指向データベースを使用するか、Amazon S3 にデータをエクスポートし、Athena または Amazon QuickSight を使用して分析を実行することを検討します。

Question	Things to consider
What level of durability does the data require?	• Aurora は、リージョン内の 3 つのアベイラビリティーゾーンにわたってデータを自動的に複製します。これはつまり、データの耐久性が高く、データ損失の可能性が低くなることを意味します。 • DynamoDB は、複数のアベイラビリティーゾーンに自動的に複製され、高可用性とデータ耐久性を提供します。 • Amazon S3 は、99.999999999% (イレブンナイン) の耐久性を備えています。Amazon RDS や DynamoDB などの多くのデータベースサービスでは、長期的な保持とアーカイブのために Amazon S3 へのデータのエクスポートをサポートしています。
Is there a desire to move away from commercial database engines or licensing costs?	• Amazon RDS または Aurora で PostgreSQL や MySQL などのオープンソースのエンジンを検討します。 • 商用データベースエンジンからオープンソースへの移行には、AWS Database Migration Service や AWS Schema Conversion Tool を利用します。
What is the operational expectation for the database? Is moving to managed services a primary concern?	• Amazon EC2 の代わりに Amazon RDS を利用し、NoSQL データベースをセルフホスティングする代わりに DynamoDB または Amazon DocumentDB を利用することで、運用上の諸経費を削減できます。

Question	Things to consider
How is the database currently accessed? Is it only application access, or are there business intelligence (BI) users and other connected off-the-shelf applications?	• 外部ツールに依存している場合は、ツールがサポートするデータベースとの互換性を維持することが必要になる場合があります。Amazon RDS は Microsoft SQL Server、Oracle、MySQL、PostgreSQL など、サポートするさまざまなエンジンバージョンとの完全な互換性があります。

- 非運用環境で実験とベンチマーキングを行い、どのデータストアがワークロード要件に対応できるかを特定します。

リソース

関連するドキュメント:

- Amazon EBS ボリュームの種類
- Amazon EC2 ストレージ
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre のパフォーマンス
- Amazon FSx for Windows File Server パフォーマンス
- Amazon S3 Glacier: S3 Glacier ドキュメント
- 設計パターンのベストプラクティス: Amazon S3 のパフォーマンスの最適化
- AWS でのクラウドストレージ
- Amazon EBS の特性とモニタリング
- AWS クラウドのデータベース
- AWS Database Caching
- DynamoDB Accelerator
- Amazon Aurora を使用する際のベストプラクティス
- Amazon Redshift パフォーマンス
- Top 10 Performance Tuning Tips for Amazon Athena
- Best Practices for Amazon Redshift Spectrum
- Amazon DynamoDB を使用した設計とアーキテクチャの設計に関するベストプラクティス

- Choose between Amazon EC2 and Amazon RDS
- Amazon ElastiCache の実装におけるベストプラクティス

関連動画:

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimizing storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2022: Building modern data architectures on AWS
- AWS re:Invent 2022: Building data mesh architectures on AWS
- AWS re:Invent 2023: Deep dive into Amazon Aurora and its innovations
- AWS re:Invent 2023: Advanced data modeling with Amazon DynamoDB
- AWS re:Invent 2022: Modernize apps with purpose-built databases
- Amazon DynamoDB deep dive: Advanced design patterns

関連する例:

- AWS Purpose Built Databases Workshop
- Databases for Developers
- AWS Modern Data Architecture Immersion Day
- Build a Data Mesh on AWS
- Amazon S3 Examples
- Optimize Data Pattern using Amazon Redshift Data Sharing
- データベースの移行
- MS SQL Server - AWS Database Migration Service (AWS DMS) Replication Demo
- Database Modernization Hands On Workshop
- Amazon Neptune Samples

PERF03-BP02 データストアで利用可能な設定オプションを評価する

データストアで使用できるさまざまな機能と設定オプションを理解して評価し、ワークロードに合わせてストレージ容量とパフォーマンスを最適化します。

一般的なアンチパターン:

- すべてのワークロードに対して、Amazon EBS などの 1 つのストレージタイプのみを使用する。
- すべてのストレージ層に対して実際のテストを行うことなく、すべてのワークロードにプロビジョンド IOPS を使用する。
- 選択したデータ管理ソリューションの設定オプションを把握していない。
- 使用できる設定オプションを確認せずに、インスタンスサイズを増やすことのみに頼っている。
- データストアのスケーリング特性をテストしていない。

このベストプラクティスを確立するメリット: データストアの設定を確認し、試してみることで、インフラストラクチャのコストを削減し、パフォーマンスとスケーラビリティを向上させ、ワークロードの維持に必要な労力を軽減できる場合があります。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

ワークロードには、データストレージとアクセス要件に基づいて 1 つまたは複数のデータストアを使用できます。パフォーマンス効率とコストを最適化するには、データアクセスパターンを評価し、適切なデータストア設定を判別する必要があります。データストアのオプションを検討する際には、ストレージオプション、メモリ、コンピューティング、リードレプリカ、整合性要件、接続プーリング、キャッシュオプションなど、さまざまな側面を考慮します。こうしたさまざまな設定オプションを試し、パフォーマンス効率のメトリクスを改善します。

実装手順

- データストアの現在の設定 (インスタンスタイプ、ストレージサイズ、データベースエンジンのバージョンなど) を把握します。
- AWS ドキュメントとベストプラクティスで、データストアのパフォーマンス向上に推奨される設定オプションについて確認します。考慮すべき主なデータストアのオプションは次のとおりです。

Configuration option	Examples
Offloading reads (like read replicas and caching)	• DynamoDB テーブルの場合、キャッシュに DAX を使用して読み取りをオフロードできます。 • Amazon ElastiCache for Redis クラスターを作成し、アプリケーションで最初にキャッシュから読み取り、要求されたアイテムが存在しない場合はデータベースにフォールバックするように設定できます。 • Amazon RDS や Aurora などのリレーショナルデータベース、Neptune や Amazon DocumentDB などのプロビジョンド NoSQL データベースはすべて、ワークロードの読み取り部分をオフロードするためのリードレプリカの追加をサポートしています。 • DynamoDB などのサーバーレスデータベースは自動的にスケーリングします。ワークロードを処理するのに十分な読み取りキャパシティユニット (RCU) がプロビジョニングされていることを確認します。

Configuration option	Examples
Scaling writes (like partition key sharding or introducing a queue)	• リレーショナルデータベースの場合、インスタンスのサイズを増やして増加したワークロードに対応するか、プロビジョンド IOPS を増やして、基盤となるストレージへのスループットを増やせるようにします。 • また、データベースに直接書き込むのではなく、データベースの前にキューを導入することもできます。このパターンでは、データの取り込みをデータベースから切り離し、フローレートを制御することで、データベースが過負荷になるのを回避できます。 • 存続時間の短いトランザクションを大量に作成するのではなく、書き込みリクエストをバッチ処理することで、書き込み量の多いリレーショナルデータベースのスループットを向上させることができます。 • DynamoDB のようなサーバーレスデータベースは、キャパシティモードに応じて、自動的に、またはプロビジョニングされた書き込みキャパシティユニット (WCU) を調整することにより、書き込みスループットをスケーリングできます。 • それでも、特定のパーティションキーのスループット制限に達すると、ホットパーティションで問題が発生する可能性があります。これは、より均等に分散されたパーティションキーを選択するか、パーティションキーを書き込みシャーディングすることで緩和できます。

Configuration option	Examples
Policies to manage the lifecycle of your datasets	• Amazon S3 ライフサイクルを使用して、オブジェクトのライフサイクル全体を管理できます。アクセスパターンが不明、変化している、または予測できない場合、Amazon S3 Intelligent-Tiering を使用できます。Intelligent-Tiering は、アクセスパターンをモニタリングし、アクセスされていないオブジェクトをコストの低いアクセス層に自動的に移動します。Amazon S3 ストレージレンズのメトリクスを活用し、ライフサイクル管理における最適化の機会やギャップを特定できます。 • Amazon EFS ライフサイクル管理は、ファイルシステムのファイルストレージを自動的に管理します。
Connection management and pooling	• Amazon RDS や Aurora でデータベースへの接続を管理するには、Amazon RDS Proxy を使用できます。 • DynamoDB などのサーバーレスデータベースには関連付けられている接続はありませんが、負荷の急増に対応するためにプロビジョンドキャパシティおよび自動スケーリングのポリシーを検討します。

- 非運用環境で実験とベンチマーキングを行い、どの設定オプションがワークロード要件に対応できるかを特定します。
- 実験が終わったら、移行を計画し、パフォーマンスメトリクスを検証します。
- AWS のモニタリングツール (Amazon CloudWatch など) と最適化ツール (Amazon S3 Storage Lens など) を使用して、実際の使用パターンに基づいてデータストアを継続的に最適化します。

リソース

関連するドキュメント:

- AWS でのクラウドストレージ
- Amazon EBS ボリュームの種類
- Amazon EC2 ストレージ
- Amazon EFS: Amazon EFS Performance
- Amazon FSx for Lustre のパフォーマンス
- Amazon FSx for Windows File Server パフォーマンス
- Amazon S3 Glacier: S3 Glacier ドキュメント
- 設計パターンのベストプラクティス: Amazon S3 のパフォーマンスの最適化
- Amazon EBS の特性とモニタリング
- AWS クラウドのデータベース
- AWS Database Caching
- DynamoDB Accelerator
- Amazon Aurora を使用する際のベストプラクティス
- Amazon Redshift パフォーマンス
- Top 10 Performance Tuning Tips for Amazon Athena
- Best Practices for Amazon Redshift Spectrum
- Amazon DynamoDB を使用した設計とアーキテクチャの設計に関するベストプラクティス

関連動画:

- AWS re:Invent 2023: Improve Amazon Elastic Block Store efficiency and be more cost-efficient
- AWS re:Invent 2023: Optimize storage price and performance with Amazon Simple Storage Service
- AWS re:Invent 2023: Building and optimizing a data lake on Amazon Simple Storage Service
- AWS re:Invent 2023: What's new with AWS file storage
- AWS re:Invent 2023: Dive deep into Amazon DynamoDB

関連する例:

- AWS Purpose Built Databases Workshop
- Databases for Developers
- AWS Modern Data Architecture Immersion Day

- Amazon EBS Autoscale
- Amazon S3 Examples
- Amazon DynamoDB Examples
- AWS データベース移行サンプル
- Database Modernization Workshop
- Working with parameters on your Amazon RDS for Postgress DB

PERF03-BP03 データストアのパフォーマンスメトリクスを収集・記録する

データストアに関連するパフォーマンスメトリクスを追跡して記録することで、データ管理ソリューションのパフォーマンスを把握できます。こうしたメトリクスは、データストアの最適化を行い、ワークロードの要件が満たされていることを確認し、ワークロードのパフォーマンスを明確に把握するのに役立ちます。

一般的なアンチパターン:

- メトリクスの検索に手動ログファイルのみを使用している。
- チームが使用する内部ツールにのみメトリクスを発行しており、ワークロードの全体像を把握できていない。
- 一部のモニタリングソフトウェアで記録されるデフォルトのメトリクスのみを使用している。
- 問題が発生したときにだけメトリクスを確認している。
- システムレベルのメトリクスのみをモニタリングし、データアクセスや使用状況に関するメトリクスを把握していない。

このベストプラクティスを活用するメリット: パフォーマンスのベースラインを確立すると、ワークロードの通常の動作とワークロードの要件を理解するのに役立ちます。異常なパターンをより迅速に特定してデバッグできるため、データストアのパフォーマンスと信頼性が向上します。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

データストアのパフォーマンスをモニタリングするには、一定期間にわたって複数のパフォーマンスメトリクスを記録する必要があります。これにより、異常を検出できるだけでなく、ビジネスメトリ

クスに照らしてパフォーマンスを測定して、ワークロードのニーズを満たしていることを確認できます。

メトリクスは、データストアをサポートする基盤システムとデータストア自体の両方のメトリクスが含まれている必要があります。基盤システムのメトリクスには、CPU 使用率、メモリ、使用可能なディスク容量、ディスク I/O、キャッシュヒット率、ネットワークのインバウンドとアウトバウンドに関するメトリクスなどがあり、データストアのメトリクスには 1 秒あたりのトランザクション数、上位のクエリ、平均クエリレート、応答時間、インデックス使用率、テーブルロック、クエリのタイムアウトの数、開いている接続の数などがあります。このデータは、ワークロードのパフォーマンスやデータ管理ソリューションの使用状況を理解するために不可欠です。これらのメトリクスをデータ駆動型アプローチの一部として使用し、ワークロードのリソースを調整および最適化します。

データベースのパフォーマンスに関連するパフォーマンスの測定値を記録するツール、ライブラリ、システムを使用します。

実装手順

1. データストアで追跡すべき主要なパフォーマンスメトリクスを特定します。
   a. Amazon S3 のメトリクスとディメンション
   b. Amazon RDS インスタンスのモニタリングメトリクス
   c. Amazon RDS での Performance Insights を使用した DB 負荷のモニタリング
   d. Enhanced Monitoring の概要
   e. DynamoDBのメトリクスとディメンション
   f. DynamoDB Accelerator のモニタリング
   g. Amazon CloudWatch を使用した Amazon MemoryDB for Redis のモニタリング
   h. モニタリングすべきメトリクス
   i. Amazon Redshift クラスターのパフォーマンスのモニタリング
   j. Timestream のメトリクスとディメンション
   k. Amazon Aurora での Amazon CloudWatch メトリクス
   l. Amazon Keyspaces (for Apache Cassandra) でのログ記録とモニタリング
   m. Amazon Neptune リソースのモニタリング
2. 承認されたロギングおよびモニタリングソリューションを使用して、これらのメトリクスを収集します。 Amazon CloudWatch では、アーキテクチャ内のリソース全体のメトリクスを収集できます。また、カスタムメトリクスを収集および発行して、ビジネスメトリクスまたは導出メトリ

クスを表面化することも可能です。CloudWatch またはサードパーティーのソリューションを使用して、しきい値を超過したことを示すアラームを設定します。

3. データストアのモニタリングに、パフォーマンスの異常を検出する機械学習ソリューションが役立つかどうかを確認します。
   a. Amazon DevOps Guru for Amazon RDS は、パフォーマンス上の問題を可視化し、是正措置についての推奨事項を提供します。
4. セキュリティと運用の目標に合わせて、モニタリングおよびログ記録ソリューションのデータ保持を設定します。
   a. CloudWatch メトリクスのデフォルトのデータ保持
   b. CloudWatch Logs のデフォルトのデータ保持

リソース

関連ドキュメント:

- AWS Database Caching (AWS データベースキャッシング)
- Amazon Athena top 10 performance tips
- Amazon Aurora を使用する際のベストプラクティス
- DynamoDB Accelerator
- Amazon DynamoDB ベストプラクティス
- Amazon Redshift Spectrum ベストプラクティス
- Amazon Redshift performance
- AWS でのクラウドデータベース
- Amazon RDS Performance Insights

関連動画:

- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Database Performance Monitoring and Tuning with Amazon DevOps Guru for Amazon RDS
- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- AWS re:Invent 2023 - Building and optimizing a data lake on Amazon S3

- AWS re:Invent 2023 - What's new with AWS file storage
- AWS re:Invent 2023 - Dive deep into Amazon DynamoDB
- Best Practices for Monitoring Redis Workloads on Amazon ElastiCache

関連する例:

- AWS Dataset Ingestion Metrics Collection Framework
- Amazon RDS Monitoring Workshop
- AWS Purpose Built Databases Workshop

PERF03-BP04 データストアのクエリパフォーマンスを向上させるための戦略を実装する

データを最適化し、データクエリを改善する戦略を実装して、ワークロードのスケーラビリティとパフォーマンスを向上させます。

一般的なアンチパターン:

- データストア内のデータをパーティション化しない。
- データストアへのデータの格納に 1 つのファイル形式のみを使用する。
- データストアでインデックスを使用しない。

このベストプラクティスを活用するメリット: データとクエリのパフォーマンスを最適化することで、効率の向上、コストの削減、ユーザーエクスペリエンスの向上につながります。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

データ最適化とクエリチューニングはデータストアのパフォーマンス効率の重要な側面であり、クラウドワークロード全体のパフォーマンスと応答性に影響を与えます。クエリが最適化されていないと、リソースの使用量が増え、ボトルネックが発生し、データストアの全体的な効率が低下する可能性があります。

データ最適化には、効率的なデータストレージとアクセスを確保する手法がいくつかあります。これは、データストアでのクエリパフォーマンスの向上にも役立ちます。主な戦略には、データのパー

ティション化、データ圧縮、データ非正規化などがあり、ストレージとアクセスの両方でデータを最適化するのに役立ちます。

実装手順

- データストアで実行される重要なデータクエリを把握して分析します。
- データストア内で処理速度の遅いクエリを特定し、クエリプランを使用して現在の状態を把握します。
  - Amazon Redshift でのクエリプランの分析
  - Athena での EXPLAIN および EXPLAIN ANALYZE の使用
- クエリのパフォーマンスを向上させるための戦略を実装します。主な戦略には次のものがあります。
  - 列指向ファイル形式 を使用する (Parquet または ORC など)。
  - データストア内のデータを圧縮して、ストレージ容量と I/O 操作を削減する。
  - データのパーティション化によりデータを細かく分割し、データスキャン時間を短縮する。
    - Athena でのデータのパーティション化
    - パーティションとデータ分散
  - クエリでよく使用される列にデータインデックスを作成する。
  - 頻繁に実行するクエリにはマテリアライズドビューを使用する。
    - マテリアライズドビューについて
    - Amazon Redshift でのマテリアライズドビューの作成
  - クエリに適した結合操作を選択する。2 つのテーブルを結合する場合、結合の左側に大きい方のテーブルを指定し、結合の右側に小さい方のテーブルを指定します。
  - 分散キャッシュソリューションでレイテンシーを改善し、データベースの I/O 操作の数を減らす。
  - 統計の実行などの定期的なメンテナンスを実施する。
- 非運用環境で実験し、戦略をテストする。

リソース

関連するドキュメント:

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

- Amazon Athena top 10 performance tips
- AWS Database Caching
- Best Practices for Implementing Amazon ElastiCache
- Athena でのデータのパーティション化

関連動画:

- AWS re:Invent 2023 - AWS storage cost-optimization best practices
- AWS re:Invent 2022 - Performance monitoring with Amazon RDS and Aurora, featuring Autodesk
- Optimize Amazon Athena Queries with New Query Analysis Tools

関連する例:

- Amazon S3 Select - Querying data without servers or databases
- AWS Purpose Built Databases Workshop

PERF03-BP05 キャッシュを利用するデータアクセスパターンを実装する

頻繁にアクセスされるデータを高速に取得できるようにデータをキャッシュする利点が得られるアクセスパターンを実装します。

一般的なアンチパターン:

- 頻繁に変更されるデータをキャッシュする。
- あたかも永続的に保存され、常に利用できるかのように、キャッシュされたデータに依存する。
- キャッシュされたデータの一貫性が考慮されない。
- キャッシュ実装の効率をモニタリングしない。

このベストプラクティスを活用するメリット: データをキャッシュに保存すると、読み取りレイテンシー、読み取りスループット、ユーザーエクスペリエンス、全体的な効率が向上し、コストも削減されます。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

キャッシュとは、同じデータに対する今後のリクエストの処理を高速化したり効率性を向上したりするために、データを保存することを目的としたソフトウェアまたはハードウェアコンポーネントです。キャッシュに保存されたデータは、失われた場合でも、前の計算を繰り返すか、別のデータストアから取得することで再構築できます。

データキャッシュは、アプリケーション全体のパフォーマンスを向上させ、基盤となるプライマリデータソースの負担を軽減するうえで、最も効果的な戦略の 1 つです。データは、アプリケーション内の複数のレベルでキャッシュできます。例えば、 クライアント側のキャッシュと呼ばれ、リモートコールを実行するアプリケーション内のキャッシュ、または リモートキャッシュと呼ばれる、データ保存用の高速セカンダリサービスを使ってデータを保存することもできます。

クライアント側のキャッシュ

クライアント側のキャッシュを使用すると、各クライアント (バックエンドデータストアにクエリを実行するアプリケーションまたはサービス) は、独自のクエリの結果を指定された期間、ローカルに保存できます。これにより、最初にローカルのクライアントキャッシュを確認することで、ネットワーク経由でデータストアに送信されるリクエストの数を低減できます。結果がキャッシュに存在しない場合、アプリケーションはデータストアにクエリを実行し、その結果をローカルに保存できます。このパターンにより、各クライアントは可能な限り最も近い場所 (クライアント自体) にデータを保存できるため、レイテンシーを最小限に抑えることができます。また、バックエンドデータストアが使用できない場合でも、クライアントは引き続きクエリの一部を処理できるため、システム全体の可用性が向上します。

この方法の欠点の 1 つは、複数のクライアントが関係する場合、同じキャッシュデータをローカルに保存する可能性があることです。その結果、このようなクライアント間でストレージが重複して使用されることになり、データの不整合が発生します。あるクライアントがクエリの結果をキャッシュし、1 分後に別のクライアントが同じクエリを実行して別の結果を取得する場合もあります。

リモートキャッシュ

クライアント間で重複するデータの問題を解決するには、高速外部サービス、つまり リモートキャッシュを使用してクエリしたデータを保存します。ローカルデータストアをチェックする代わりに、各クライアントはバックエンドデータストアへのクエリを実行する前にリモートキャッシュをチェックします。この戦略により、クライアント間の応答の一貫性が強化され、保存されたデータの効率が向上し、ストレージスペースがクライアントとは別個にスケールされるため、キャッシュされたデータの量が増大します。

リモートキャッシュの欠点は、リモートキャッシュをチェックするために追加のネットワークホップが必要になるため、システム全体のレイテンシーが増大する可能性がある点です。クライアント側のキャッシュをリモートキャッシュと併用してマルチレベルキャッシュを行うことで、レイテンシーを短縮できます。

実装手順

1. キャッシュの利点を活用できるデータベース、API、ネットワークサービスを特定します。読み取りワークロードが高いサービス、読み取りと書き込み率が高いサービス、またはスケールするのにコストがかかるサービスなどが、キャッシュの候補となります。
   - データベースのキャッシュ
   - API キャッシュを有効にして応答性を強化する
2. アクセスパターンに最適なキャッシュ戦略の種類を特定します。
   - キャッシュ戦略
   - AWS キャッシュソリューション
3. データストアについては、 キャッシュのベストプラクティス に従います。
4. すべてのデータに対して有効期間 (TTL) などのキャッシュ無効化戦略を設定し、データの鮮度とバックエンドデータストアへの負荷の軽減の間でバランスをとります。
5. 自動接続再試行、エクスポネンシャルバックオフ、クライアント側のタイムアウト、接続プーリングなどの機能が利用できる場合は、クライアントで有効にします。これにより、パフォーマンスと信頼性が向上します。
   - ベストプラクティス: Redis クライアントと Amazon ElastiCache for Redis
6. 80% 以上を目標にキャッシュヒットレートをモニタリングします。値が低い場合は、キャッシュサイズが不十分であるか、キャッシュの利点を活用できないアクセスパターンである可能性があります。
   - モニタリングすべきメトリクス
   - Amazon ElastiCache での Redis ワークロードモニタリングのベストプラクティス
   - Amazon CloudWatch を使用した Amazon ElastiCache for Redis のモニタリングのベストプラクティス
7. <!--ATMS sidestep. Remove this--> データレプリケーション を実装して読み取りを複数のインスタンスにオフロードし、データ読み取りのパフォーマンスと可用性を向上させます。

リソース

関連するドキュメント:

- Amazon ElastiCache Well-Architected レンズの使用
- Amazon CloudWatch を使用した Amazon ElastiCache for Redis のモニタリングのベストプラクティス
- モニタリングすべきメトリクス
- 「Amazon ElastiCache を使用した大規模環境でのパフォーマンス」ホワイトペーパー
- キャッシングの課題と戦略

関連動画:

- Amazon ElastiCache ラーニングパス
- Amazon ElastiCache 設計のベストプラクティス
- AWS re:Invent 2020 - Design for success with Amazon ElastiCache best practices
- AWS re:Invent 2023 - [LAUNCH] Introducing Amazon ElastiCache Serverless
- AWS re:Invent 2022 - 5 great ways to reimagine your data layer with Redis
- AWS re:Invent 2021 - Deep dive on Amazon ElastiCache for Redis

関連する例:

- Amazon ElastiCache for Redis を使用したMySQL データベースのパフォーマンス向上

ネットワークとコンテンツ配信

ワークロードに最適なネットワークソリューションは、レイテンシー、スループット要件、ジッター、および帯域幅に応じて異なります。ロケーションのオプションは、ユーザーまたはオンプレミスのリソースなどの物理的な制約に左右されます。これらの制約は、エッジロケーションまたはリソースの配置で相殺することができます。

AWS ではネットワーキングが仮想化されており、数多くの異なるタイプと設定で利用することができます。これにより、ネットワークのニーズへの適合が容易になります。AWS は、ネットワークトラフィックを最適化する製品機能を提供します (拡張ネットワーキング、Amazon EC2 ネットワーク最適化インスタンス、Amazon S3 Transfer Acceleration、動的 Amazon CloudFront など)。また、AWS は、ネットワーク距離またはジッターを軽減するネットワーク機能も提供します (Amazon Route 53 レイテンシールーティング、Amazon VPC エンドポイント、AWS Direct Connect、AWS Global Accelerator など)。

この重点分野では、クラウドで効率的なネットワークおよびコンテンツ配信ソリューションを設計、構成、運用するためのガイダンスとベストプラクティスを紹介します。

ベストプラクティス

- PERF04-BP01 ネットワークがパフォーマンスに与える影響を理解する
- PERF04-BP02 使用可能なネットワーク機能を評価する
- PERF04-BP03 ワークロードに適した専用接続または VPN を選択する
- PERF04-BP04 ロードバランシングを使用してトラフィックを複数のリソースに分散する
- PERF04-BP05 パフォーマンスを高めるネットワークプロトコルを選択する
- PERF04-BP06 ネットワーク要件に基づいてワークロードのロケーションを選択する
- PERF04-BP07 メトリクスに基づいてネットワーク設定を最適化する

PERF04-BP01 ネットワークがパフォーマンスに与える影響を理解する

ネットワーク関連の意思決定がワークロードに与える影響を分析して理解し、効率的なパフォーマンスとユーザーエクスペリエンスの向上を実現します。

一般的なアンチパターン:

- すべてのトラフィックが既存のデータセンターを通過する。
- クラウドネイティブなネットワークセキュリティツールを使用せずに、すべてのトラフィックを中央のファイアウォール経由でルーティングする。
- 実際の使用要件を理解せずに AWS Direct Connect 接続をプロビジョニングする。
- ワークロードの特性および暗号化にかかるコストを考慮しない。
- クラウドのネットワーク戦略にオンプレミスのコンセプトと戦略を使用する。

このベストプラクティスを活用するメリット: ネットワークがワークロードのパフォーマンスに与える影響を理解することで、潜在的なボトルネックの特定、ユーザーエクスペリエンスの改善、信頼性の向上を実現しながら、ワークロードの変化に伴う運用保守業務を軽減できます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

ネットワークは、アプリケーションコンポーネント、クラウドサービス、エッジネットワーク、オンプレミスデータ間の接続を担っているため、ワークロードのパフォーマンスに大きな影響を与える可能性があります。ワークロードのパフォーマンスに加え、ユーザーエクスペリエンスも、ネットワークのレイテンシー、帯域幅、プロトコル、場所、ネットワークの混雑、ジッター、スループット、ルーティングルールの影響を受ける可能性があります。

レイテンシー、パケットサイズ、ルーティングルール、プロトコル、サポートするトラフィックパターンなど、ワークロードのネットワーク要件をまとめて文書化します。利用可能なネットワークソリューションを確認し、ワークロードのネットワーク特性に適合するサービスを特定します。クラウドベースのネットワークは迅速に再構築できるため、パフォーマンス効率を向上させるためにもネットワーク アーキテクチャを時間とともに進化させる必要があります。

実装手順:

1. ネットワークのレイテンシー、帯域幅、プロトコル、場所、トラフィックパターン (急増とその頻度)、スループット、暗号化、点検、ルーティングルールなどのメトリクスを含め、ネットワークパフォーマンス要件を定義し、文書化します。
2. AWS の主要なネットワークサービス (VPC、AWS Direct Connect、Elastic Load Balancing (ELB)、 Amazon Route 53など) について学びます。
3. 次の主要なネットワーク特性を把握します。

特性	ツールとメトリクス
基礎的なネットワーク特性	• VPC フローログ • AWS Transit Gateway フローログ • AWS Transit Gateway のメトリクス • AWS PrivateLink のメトリクス
アプリケーションネットワークの特性	• Elastic Fabric Adapter • AWS App Mesh のメトリクス • Amazon API Gateway のメトリクス
エッジネットワークの特性	• Amazon CloudFront のメトリクス • Amazon Route 53 のメトリクス • AWS Global Accelerator のメトリクス
ハイブリッドネットワークの特性	• AWS Direct Connect のメトリクス • AWS Site-to-Site VPN のメトリクス • AWS Client VPN のメトリクス • AWS クラウド WAN のメトリクス
セキュリティネットワークの特性	• AWS Shield、AWS WAF、AWS Network Firewall のメトリクス
トレーシングの特性	• AWS X-Ray • VPC Reachability Analyzer • Network Access Analyzer • Amazon Inspector • Amazon CloudWatch RUM

4. ネットワークパフォーマンスをベンチマーキングし、テストします。
   a. ネットワークループットをベンチマーキング する: インスタンスが同じ VPC 内にある場合、いくつかの要因が Amazon EC2 のネットワークパフォーマンスに影響する可能性があるため。同じ VPC 内で Amazon EC2 Linux インスタンス間のネットワーク帯域幅を測定します。
   b. 負荷 テスト を実行し、ネットワークソリューションとオプションを試します。

リソース

関連するドキュメント:

- Application Load Balancer
- Linux での EC2 拡張ネットワーキング
- Windows での EC2 拡張ネットワーキング
- EC2 プレイスメントグループ
- Linux インスタンスで Elastic Network Adapter (ENA) を使用して拡張ネットワークを有効にする
- Network Load Balancer
- AWS ネットワークとコンテンツ配信
- Transit Gateway
- Amazon Route 53 でレイテンシーベースルーティングへ移行する
- VPC エンドポイント

関連動画:

- AWS re:Invent 2023 - AWS networking foundations
- AWS re:Invent 2023 - What can networking do for your application?
- AWS re:Invent 2023 - Advanced VPC designs and new capabilities
- AWS re:Invent 2023 - A developer's guide to cloud networking
- AWS re:Invent 2019 - Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2019 - Optimizing Network Performance for Amazon EC2 Instances
- AWS Summit Online - Improve Global Network Performance for Applications
- AWS re:Invent 2020 - Networking best practices and tips with the Well-Architected Framework
- AWS re:Invent 2020 - AWS networking best practices in large-scale migrations

関連する例:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops
- Hands-on Network Firewall Workshop

- Observing and Diagnosing your Network on AWS
- Finding and addressing Network Misconfigurations on AWS

PERF04-BP02 使用可能なネットワーク機能を評価する

パフォーマンスの向上に役立つ可能性のあるクラウドのネットワーク機能を評価します。これらの機能の影響を、テスト、メトリクス、および分析を使って測定してください。例えば、レイテンシー、ネットワーク距離、またはジッターを低減するために利用できるネットワークレベルの機能を活用します。

一般的なアンチパターン:

- 本社の物理的な所在地を理由に、1 つのリージョン内に留まる。
- セキュリティグループの代わりにファイアウォールを使用してトラフィックをフィルタリングする。
- セキュリティグループ、エンドポイントポリシー、その他のクラウドネイティブ機能に頼らず、トラフィック検査のために TLS を破る。
- セキュリティグループではなく、サブネットベースのセグメンテーションのみを使用する。

このベストプラクティスを活用するメリット: すべてのサービス機能とオプションを評価することにより、ワークロードパローマンスを向上させ、インフラストラクチャのコストを削減し、ワークロードを維持するために必要な労力を減らし、全体的なセキュリティ体制を強化できます。グローバルな AWS のバックボーンを活用すれば、最適なネットワークエクスペリエンスを顧客に提供することができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

AWS は、 AWS Global Accelerator および Amazon CloudFront などのサービスを提供します。これはネットワークパフォーマンスの向上に役立ちますが、ほとんどの AWS サービスには製品機能 (Amazon S3 Transfer Acceleration 機能) があり、ネットワークトラフィックを最適化します。

ネットワーク関連の設定オプションにはどのようなものがあるか、またそれらがワークロードにどのような影響を与えるかを確認します。パフォーマンスの最適化は、これらのオプションがアーキテクチャとどのように相互作用し、測定されたパフォーマンスとユーザーエクスペリエンスの両方に与える影響をいかに理解できるかにかかっています。

実装手順

- ワークロードコンポーネントのリストを作成します。
  - 統一されたグローバルネットワークを構築する場合は、 AWS クラウド WAN を使用して組織のネットワークを構築、管理、監視することを検討します。
  - グローバルネットワークとコアネットワークを Amazon CloudWatch Logs のメトリクスでモニタリングします。ユーザーのデジタルエクスペリエンスを特定、把握、強化するために役立つインサイトを提供する Amazon CloudWatch RUM を活用します。
  - AWS リージョン とアベイラビリティーゾーン、各アベイラビリティーゾーン内の合計ネットワークレイテンシーを AWS Network Manager を使用して表示することで、アプリケーションのパフォーマンスと基礎となる AWS ネットワークのパフォーマンスとの関係を把握できます。
  - 既存の構成管理データベース (CMDB) ツールまたはサービス (AWS Config など) を使用して、ワークロードや設定方法のインベントリを作成します。
- これが既存のワークロードである場合、ボトルネックや改善すべき領域に特化して、パフォーマンスメトリクスのベンチマークを特定し、文書化します。パフォーマンスに関連するネットワークメトリクスは、ビジネス要件やワークロード特性により、ワークロードごとに異なります。最初のうちは、帯域幅、レイテンシー、パケットロス、ジッター、再送信などのメトリックスを確認することが重要かもしれません。
- これが新しいワークロードである場合は、 負荷テスト を実施して、パフォーマンスのボトルネックを特定します。
- 特定したパフォーマンスのボトルネックに対して、ソリューションの設定オプションを確認し、パフォーマンス改善の機会を見つけます。次の主要なネットワークオプションと特徴を確認してください:

改善の機会	ソリューション
ネットワークパスまたはルート	パスまたはルートの識別には、 Network Access Analyzer を使用します。
ネットワークプロトコル	参照先: PERF04-BP05 パフォーマンスを高めるネットワークプロトコルを選択する
ネットワークトポロジ	複数のアカウントに接続する際は、 VPC ピアリング および AWS Transit Gateway の運用上およびパフォーマンス上のトレードオフを評価します。AWS Transit Gateway は、数千の

改善の機会	ソリューション
	AWS アカウントやオンプレミスネットワークにまたがるすべての VPC を相互接続する方法を簡素化します。複数のアカウント間で AWS Transit Gateway を共有するには、 AWS Resource Access Manager を使用します。 参照先: PERF04-BP03 ワークロードに適した専用接続または VPN を選択する
ネットワークサービス	AWS Global Accelerator は、AWS グローバルネットワークインフラストラクチャを使用して、ユーザーのトラフィックのパフォーマンスを最大 60% 向上するネットワークサービスです。 Amazon CloudFront は、ワークロードのコンテンツ配信のパフォーマンスとレイテンシーをグローバルに向上させることができます。 CloudFont が配信するコンテンツをでカスタマイズする関数をよりユーザーの近く実行する Lambda@Edge を使用して、レイテンシーを削減し、パフォーマンスを向上させることができます。 Amazon Route 53 では、 レイテンシーベースルーティング、位置情報ルーティング、地理的近接性ルーティング、および IP ベースルーティング のオプションを提供しており、全世界ユーザーに対するワークロードのパフォーマンスを向上することができます。ワークロードがグローバルに分散している場合に、ワークロードトラフィックとユーザーロケーションを確認することにより、どのルーティングオプションによってワークロードパフォーマンスが最適化されるかを特定できます。

改善の機会	ソリューション
ストレージリソース機能	Amazon S3 Transfer Acceleration により、外部ユーザーは、Amazon S3 にデータをアップロードする際に CloudFront のネットワーク最適化機能のメリットを享受できます。これにより、AWS クラウド への専用接続がないリモートロケーションから大量のデータを転送する機能が向上します。 Amazon S3 マルチリージョンアクセスポイント は、1 つのアクセスポイントを提供することにより、複数のリージョンへコンテンツをレプリケートし、ワークロードを簡素化します。マルチリージョンアクセスポイントが使用されている場合、低レイテンシーバケットを特定するサービスによって、データを要求したり Amazon S3 に書き込むことができます。

改善の機会	ソリューション
コンピューティングリソース機能	Elastic Network Interfaces (ENA) は Amazon EC2 インスタンス、コンテナ、および Lambda 関数によって使用され、フロー単位で制限されています。プレイスメントグループを見直して、EC2 ネットワークスループットを最適化します。フロー単位でのボトルネックを避けるために、複数フローを使用できるようアプリケーションを設計します。コンピューティング関連のネットワークメトリクスをモニタリングおよび可視化するには、CloudWatch と ethtool を使用します。この `ethtool` コマンドは ENA ドライバーに含まれており、追加のネットワーク関連のメトリクスを公開します。これらは、カスタムメトリクス として CloudWatch に発行できます。 Amazon Elastic Network Adapters (ENA) は、クラスタープレイスメントグループ内のインスタンスのスループットを向上させることで、さらなる最適化を実現します。 Elastic Fabric Adapter (EFA) は、高いレベルのインターノードコミュニケーションを AWS で大規模に要求するワークロードの実行を可能にする、Amazon EC2 インスタンスのネットワークインターフェイスです。 Amazon EBS 最適化インスタンス は、最適化された設定スタックを使用し、Amazon EBS I/O を増加させるために専用の追加容量を提供します。

リソース

関連するドキュメント:

- Application Load Balancer
- Linux での EC2 拡張ネットワーキング
- Windows での EC2 拡張ネットワーキング
- EC2 プレイスメントグループ
- Linux インスタンスで Elastic Network Adapter (ENA) を使用して拡張ネットワークを有効にする
- Network Load Balancer
- AWS ネットワークとコンテンツ配信
- Amazon Route 53 でレイテンシーベースルーティングへ移行する
- VPC エンドポイント
- VPC フローログ

関連動画:

- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities
- AWS re:Invent 2023 – A developer's guide to cloud networking
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure
- AWS re:Invent 2019 – Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2018 – Optimizing Network Performance for Amazon EC2 Instances
- AWS Global Accelerator

関連する例:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops
- Observing and diagnosing your network
- Finding and addressing network misconfigurations on AWS

PERF04-BP03 ワークロードに適した専用接続または VPN を選択する

オンプレミスのリソースとクラウドのリソースを接続するためにハイブリッド接続が必要な場合は、パフォーマンス要件を満たす十分な帯域幅をプロビジョニングします。ハイブリッドワークロードの帯域幅とレイテンシーの要件を見積もります。これらの値によってサイズ要件が決まります。

一般的なアンチパターン:

- ネットワークの暗号化要件の VPN ソリューションのみを評価する。
- バックアップ接続または冗長接続の選択肢を評価しない。
- ワークロードのすべての要件 (暗号化、プロトコル、帯域幅、トラフィックのニーズ) を特定しない。

このベストプラクティスを活用するメリット: 適切な接続ソリューションを選択して構成することで、ワークロードの信頼性を高め、パフォーマンスを最大化できます。ワークロード要件を特定して事前計画を行い、ハイブリッドソリューションを評価することで、時間対価値を高めながら、コストの高いネットワークの物理的変更と運用上の諸経費を最小限に抑えることができます。

このベストプラクティスを活用しない場合のリスクレベル: 高

実装のガイダンス

帯域幅要件に基づいてハイブリッドネットワーキングアーキテクチャを開発します。AWS Direct Connect を使用すると、オンプレミスネットワークを AWS にプライベート接続できます。一貫したパフォーマンスを達成しながら、高帯域幅、低レイテンシーが求められる場合に適しています。VPN 接続は、インターネット上で安全な接続を確立します。一時的な接続のみが必要な場合、コストが重要な場合、または AWS Direct Connect 使用時に耐障害性のある物理ネットワーク接続が確立されるのを待つ間、不測の事態に備えて使用されます。

帯域幅の要件が高い場合は、AWS Direct Connect または VPN サービスを複数使用することを検討します。トラフィックはサービス間で負荷分散できますが、レイテンシ–と帯域幅の違いから、AWS Direct Connect と VPN 間でのロードバランシングはお勧めしません。

実装手順

1. 既存のアプリケーションの帯域幅とレイテンシーの要件を見積もります。

   a. AWS に移行している既存のワークロードについては、内部ネットワークモニタリングシステムからのデータを活用します。

   b. モニタリングデータがない新規または既存のワークロードついては、プロダクトオーナーと相談のうえ、適切なパフォーマンスメトリクスを導き出し、優れたユーザーエクスペリエンスを提供します。

2. 接続オプションとして専用接続または VPN 接続を選択します。すべてのワークロード要件 (暗号化、帯域幅、トラフィックのニーズ) に基づいて、AWS Direct Connect または AWS VPN (あるいは両方) を選択できます。下の図は、適切な接続タイプの選択に役立ちます。

   a. AWS Direct Connect は、専用接続またはホスト接続を使用して、50 Mbps から 100 Gbps の AWS 環境への専用接続を提供します。これにより、管理および制御されたレイテンシーとプロビジョニングされた帯域幅が提供され、ワークロードが効率の良い方法でその他の環境に接続できます。AWS Direct Connect パートナーを使用すると、複数の環境からエンドツーエンドの接続を確立し、一貫性あるパフォーマンスを備えた拡張ネットワークを実現できます。AWS を使用すると、ネイティブ 100 Gbps、Link Aggregation Group (LAG)、または BGP Equal-cost multipath (ECMP) のいずれかを使用して、直接接続の帯域幅をスケールできます。

   b. AWS の Site-to-Site VPN は、インターネットプロトコルセキュリティ (IPsec) をサポートするマネージド VPN サービスを提供します。VPN 接続が作成されると、高可用性に向けて 各 VPN 接続に 2 つのトンネルが含まれています。

3. AWS のドキュメントに従って、適切な接続オプションを選択します。

   a. AWS Direct Connect を使用する場合は、接続に適した帯域幅を選択してください。

   b. 複数の口ケーションで AWS Site-to-Site VPN を使用して AWS リージョン に接続している場合は、 高速 Site-to-Site VPN 接続 を使用すると、ネットワークのパフォーマンスが向上する可能性があります。

   c. ネットワーク設計が AWS Direct Connect を使用した IPSec VPN 接続で構成されている場合は、プライベート IP VPN を使用してセキュリティを強化し、セグメンテーションを実現することを検討してください。 AWS サイト間プライベート IP VPN は、トランジット仮想インターフェイス (VIF) の上にデプロイされます。

   d. AWS Direct Connect SiteLink では、データを AWS Direct Connect口ケーション間の最速の経路で AWS リージョン をバイパスして送信することで、世界中のデータセンター間で低レイテンシ–の冗長な接続を確立できます。

4. 本番環境にデプロイする前に、接続設定を検証します。セキュリティとパフォーマンスのテストを実施して、帯域幅、信頼性、レイテンシ–、コンプライアンスの要件を満たしていることを確認します。

5. 接続のパフォーマンスと使用状況を定期的に監視し、必要に応じて最適化します。

決定論的なパフォーマンスについてのフローチャート

リソース

関連するドキュメント:

- AWS ネットワークとコンテンツ配信
- AWS Transit Gateway
- VPC エンドポイント
- スケーラブルで安全なマルチ VPC の AWS ネットワークインフラストラクチャを構築する
- Client VPN

関連動画:

- AWS re:Invent 2023 – Building hybrid network connectivity with AWS
- AWS re:Invent 2023 – Secure remote connectivity to AWS
- AWS re:Invent 2022 – Optimizing performance with Amazon CloudFront
- AWS re:Invent 2019 – Connectivity to AWS and hybrid AWS network architectures
- AWS re:Invent 2020 – AWS Transit Gateway Connect

関連する例:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF04-BP04 ロードバランシングを使用してトラフィックを複数のリソースに分散する

トラフィックを複数のリソースやサービスに分散させ、ワークロードがクラウドの伸縮性を活用できるようにします。また、ロードバランシングを使用して暗号化ターミネーションをオフロードし、パフォーマンスと信頼性を向上させ、トラフィックを効率的に管理およびルーティングすることもできます。

一般的なアンチパターン:

- ロードバランサーの種類を選択する際にワークロードの要件を考慮していない。
- パフォーマンスの最適化のためにロードバランサー機能を活用していない。
- ワークロードがロードバランサーなしで直接インターネットに公開されている。
- 既存のロードバランサーを介して、すべてのインターネットトラフィックをルーティングしている。
- 汎用 TCP ロードバランシングを使用して、各コンピューティングノードが SSL 暗号化を処理するようにしている。

このベストプラクティスを確立するメリット: ロードバランサーは、単一のアベイラビリティーゾーン内または複数のアベイラビリティーゾーンに渡るアプリケーショントラフィックのさまざまな負荷を処理し、ワークロードの高可用性、自動スケーリング、効率的な使用を可能にします。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

ロードバランサーは、ワークロードのエントリポイントとして機能し、そこからコンピューティングインスタンスやコンテナなどのバックエンドターゲットにトラフィックを分散して使用率を改善します。

適切なロードバランサーの種類を選択することが、アーキテクチャを最適化するうえでの最初のステップとなります。まず、プロトコル (TCP、HTTP、TLS、または WebSocket など)、ターゲットの種類 (インスタンス、コンテナ、またはサーバーレスなど)、アプリケーション要件 (長時間実行される接続、ユーザー認証、またはスティッキーなど) や配置 (リージョン、ローカルゾーン、アウトポスト、ゾーン分離など) のワークロードの特性をリストアップすることから始めます。

AWS は、アプリケーションでロードバランシングを使用するためのモデルを複数提供しています。Application Load Balancer は、HTTP トラフィックと HTTPS トラフィックのロードバランシングに最適です。マイクロサービスやコンテナといったモダンアプリケーションアーキテクチャを対象とした高度なリクエストルーティングを実現できます。

Network Load Balancer は、きわめて高いパフォーマンスが要求される TCP トラフィックのロードバランシングに最適です。超低レイテンシーを維持しながら 1 秒あたり数百万件ものリクエストを処理することができ、突発的、または変動しやすいトラフィックパターンを処理するために最適化されています。

Elastic Load Balancing は、統合された証明書管理と SSL/TLS 復号化を提供するため、ロードバランサーの SSL 設定を一元的に管理し、CPU に負荷のかかる SSL/TLS 処理をお客様のワークロードからオフロードすることができます。

適切なロードバランサーを選択したら、ロードバランサーの機能を活用して、バックエンドが必要とするトラフィック処理のための労力を削減できます。

例えば、Application Load Balancer (ALB) と Network Load Balancer (NLB) の両方を使用して、SSL/TLS 暗号化オフロードを実行できます。これにより、CPU 負荷が高い TLS ハンドシェイクのターゲットによる終了を回避して、証明書管理を改善する機会が得られます。

ロードバランサーで SSL/TLS オフロードを設定すると、暗号化されていないトラフィックをバックエンドに配信すると同時に、クライアントとの間のトラフィックの暗号化の役目を果たすため、バックエンドリソースが解放され、クライアントにとっての応答時間が改善されます。

Application Load Balancer は、ターゲットでのサポートを必要とせずに HTTP2 トラフィックを処理することもできます。このようなシンプルな決断をすることで、HTTP2 は TCP 接続をより効率的に使用するようになり、アプリケーションの応答時間を改善できます。

アーキテクチャを定義する際は、ワークロードのレイテンシー要件を考慮する必要があります。例えば、レイテンシーの影響を受けやすいアプリケーションがある場合、非常に低レイテンシーを実現する Network Load Balancer を使用するよう決定することができます。または、AWS Local Zones、あるいは AWS Outposts で Application Load Balancer を活用して、ワークロードをより顧客の近くに配置する方法を採用することもできます。

レイテンシーの影響を受けやすいワークロードに関して考慮すべきもう 1 つの点は、クロスゾーン負荷分散です。クロスゾーン負荷分散を使用すると、各ロードバランサーノードは許可されたすべてのアベイラビリティーゾーン内の登録済みターゲットにトラフィックを分散します。

ロードバランサーと統合された Auto Scaling を使用します。パフォーマンス効率に優れたシステムの重要な側面の 1 つに、バックエンドリソースのサイズの適切な設定があります。これを実現するには、バックエンドターゲットリソースのロードバランサー統合を利用できます。Auto Scaling グループと統合したロードバランサーを使用することで、受信トラフィックに対応して、必要に応じてターゲットがロードバランサーに追加されたり削除されたりします。また、コンテナ化されたワークロードには、ロードバランサーを Amazon ECS および Amazon EKS と統合できます。

- Amazon ECS - Service load balancing
- Application load balancing on Amazon EKS
- Network load balancing on Amazon EKS

実装手順

- トラフィック量、可用性、アプリケーションのスケーラビリティなど、ロードバランシングの要件を定義します。
- アプリケーションに適したロードバランサータイプを選択します。
  - HTTP/HTTPS のワークロードには、Application Load Balancer を使用します。
  - TCP または UDP で実行される HTTP 以外のワークロードには、Network Load Balancer を使用します。
  - 両方の製品の機能を活用したい場合は、両方を組み合わせて (ALB を NLB のターゲットとして) 使用します。例えば、NLB の静的 IP を ALB からの HTTP ヘッダーベースのルーティングと組み合わせて使用する場合や、HTTP のワークロードを AWS PrivateLink に公開する場合に使用します。
  - すべてのロードバランサー製品の比較については、「ELB の製品の比較」を参照してください。
- 可能であれば SSL/TLS オフロードを使用します
  - AWS Certificate Manager と統合された Application Load Balancer と Network Load Balancer の両方で HTTPS/TLS リスナーを設定します。
  - ワークロードによっては、コンプライアンス上の理由で、エンドツーエンドの暗号化が必要になる場合があることに注意します。この場合は、ターゲットで暗号化を許可する必要があります。
  - セキュリティのベストプラクティスについては、「SEC09-BP02 Enforce encryption in transit」を参照してください。
- 適切なルーティングアルゴリズムを選択します (ALB のみ)。
  - ルーティングアルゴリズムは、バックエンドターゲットの使用状況に変化をもたらし、パフォーマンスへの影響を左右します。例えば、ALB には、以下の 2 種類のルーティングアルゴリズムが用意されています。
  - 最小未処理リクエスト: アプリケーションのリクエストの複雑性が異なる場合や、ターゲットの処理能力が異なる場合に、バックエンドターゲットへのロードバランシングを改善するために使用します。
  - ラウンドロビン: リクエストとターゲットが同様の場合、またはリクエストをターゲット間で均等に分散する必要がある場合に使用します。
- クロスゾーンまたはゾーン分離を検討します。
  - レイテンシーの改善とゾーンの障害ドメイン対策として、クロスゾーンをオフ (ゾーン分離) で使用します。NLB ではデフォルトでオフになっていますが、ALB ではターゲットグループごとにオフに設定できます。

    - 可用性と柔軟性の向上のために、クロスゾーンをオンにします。ALB ではデフォルトでクロスゾーンはオンになっています。NLB ではターゲットグループごとにオンに設定できます。
- HTTP ワークロードの HTTP キープアライブをオンにします (ALB のみ)。この機能を使用すると、ロードバランサーはキープアライブタイムアウトが期限切れになるまでバックエンド接続を再利用できるため、HTTP リクエストと応答時間が改善され、バックエンドターゲットでのリソース使用率も低減します。Apache と Nginx でこれを実行する方法の詳細については、「ELB のバックエンドサーバーとして Apache または NGINX を使用するための最適な設定を教えてください」を参照してください。
- ロードバランサーのモニタリングをオンにします。
    - Application Load Balancer と Network Load Balancer のアクセスログを有効にします。
    - ALB で考慮すべき主なフィールドは、`request_processing_time`、`request_processing_time`、`response_processing_time` です。
    - NLB で考慮すべき主なフィールドは、`connection_time` と `tls_handshake_time` です。
    - 必要な際にログをクエリできるように準備を整えておきます。Amazon Athena を使用すると、ALB ログと NLB ログの両方をクエリできます。
    - ALB の `TargetResponseTime` などのパフォーマンス関連のメトリクスのアラームを作成します。

リソース

関連するドキュメント:

- ELB の製品の比較
- AWS グローバルインフラストラクチャ
- Improving Performance and Reducing Cost Using Availability Zone Affinity
- Step by step for Log Analysis with Amazon Athena
- Querying Application Load Balancer logs
- Monitor your Application Load Balancers
- Monitor your Network Load Balancer
- Use Elastic Load Balancing to distribute traffic across the instances in your Auto Scaling group

関連動画:

- AWS re:Invent 2023: What can networking do for your application?
- AWS re:Inforce 20: How to use Elastic Load Balancing to enhance your security posture at scale
- AWS re:Invent 2018: Elastic Load Balancing: Deep Dive and Best Practices
- AWS re:Invent 2021 - How to choose the right load balancer for your AWS workloads
- AWS re:Invent 2019: Get the most from Elastic Load Balancing for different workloads

関連する例:

- Gateway Load Balancer
- CDK and AWS CloudFormation samples for Log Analysis with Amazon Athena

PERF04-BP05 パフォーマンスを高めるネットワークプロトコルを選択する

ワークロードのパフォーマンスに対する影響に基づいて、システムとネットワーク間における通信のためのプロトコルを決定します。

スループットの達成には、レイテンシーと帯域幅間の関係が関与します。ファイル転送で Transmission Control Protocol (TCP) を使用している場合は、レイテンシーが高くなると全体的なスループットを低下させる可能性が高くなります。これを解決するアプローチには、TCP チューニングと最適化された転送プロトコルを使うものもありますが、1 つの解決策として User Datagram Protocol (UDP) を使用する方法があります。

一般的なアンチパターン:

- パフォーマンス要件に関係なく、すべてのワークロードに TCP を使用する。

このベストプラクティスを活用するメリット: ユーザーとワークロードコンポーネント間の通信に適切なプロトコルが使用されていることを確認すると、アプリケーションのユーザーエクスペリエンスが全体的に向上します。例えば、コネクションレス型 UDP では高速通信が可能ですが、再送信や高い信頼性は提供されません。TCP はフル機能のプロトコルですが、パケットの処理にはより大きなオーバーヘッドが必要です。

このベストプラクティスを活用しない場合のリスクレベル: 中

実装のガイダンス

アプリケーションに合わせて異なるプロトコルを選択する能力があり、この分野の専門知識を持っている場合は、異なるプロトコルを使用してアプリケーションとエンドユーザーエクスペリエンスを最適化してください。このアプローチは非常に難しいため、先に他の方法でアプリケーションを最適化したことがある場合にのみ試してください。

レイテンシーとスループットの要件を理解し、パフォーマンスを最適化するネットワークプロトコルを選択することが、ワークロードのパフォーマンスを向上するうえで重要となる考慮事項です。

TCP の使用を検討するべきケース

TCP は信頼性に優れたデータ配信を提供し、データの信頼性と確実な配信が重要となるワークロードのコンポーネント間の通信に利用できます。多くのウェブベースのアプリケーションは、HTTP や HTTPS などの TCP ベースのプロトコルを使用して TCP ソケットを開き、アプリケーションコンポーネント間の通信を行っています。TCP はアプリケーションコンポーネント間のシンプルで信頼性の高い転送メカニズムであるため、メールやファイルのデータ転送などのアプリケーションでも一般的に TCP が利用されます。TCP で TLS を使用すると通信のオーバーヘッドが増加し、レイテンシーが増加して、スループットが低下する可能性がありますが、セキュリティ上のメリットがあります。このオーバーヘッドは主にハンドシェークプロセスの追加オーバーヘッドにより発生し、完了するまでに数回のラウンドトリップが必要になる場合があります。ハンドシェイクが完了すると、データの暗号化と復号化のオーバーヘッドは比較的少なくなります。

UDP の使用を検討するべきケース

UDP はコネクションレス型のプロトコルであるため、ログ、モニタリング、VoIP データなどの、高速かつ効率的な転送が必要なアプリケーションに適しています。また、多数のクライアントからの小型のクエリに対応するワークロードのコンポーネントがある場合は、ワークロードの最適なパフォーマンスを確保するために UDP の使用を検討します。UDP では、Datagram Transport Layer Security (DTLS) が Transport Layer Security (TLS) に相当します。UDP で DTLS を使用する場合、ハンドシェイクプロセスは簡素化され、オーバーヘッドはデータの暗号化と復号化から発生します。また、DTLS には、セキュリティパラメータを提示し、改ざんを検出するための追加フィールドが含まれているため、UDP パケットに少量のオーバーヘッドが追加されます。

SRD の使用を検討するべきケース

Scalable Reliable Datagram (SRD) は、複数のパス間でトラフィックの負荷分散を行い、パケットドロップやリンク障害から迅速に回復できる機能を備えており、高スループットのワークロード向けに最適化されたネットワークトランスポートプロトコルです。そのため、SRD は、コンピューティン

グノード間で高スループットと低レイテンシー通信を必要とするハイパフォーマンスコンピューティング (HPC) ワークロードに最適です。ノード間で大量のデータ転送を伴うシミュレーション、モデリング、データ分析などの並列処理タスクなどで利用される場合があります。

実装手順

1. オンラインファイル転送アプリケーションのスループット向上のために、 AWS Global Accelerator および AWS Transfer Family サービスを使用します。AWS Global Accelerator サービスを使用すると、クライアントデバイスと AWS 上のワークロード間のレイテンシーが低減されます。AWS Transfer Family では、Secure Shell File Transfer Protocol (SFTP) と File Transfer Protocol over SSL (FTPS) などの TCP ベースのプロトコルを使用して、AWS ストレージサービスへのファイル転送を安全にスケールおよび管理できます。
2. ワークロードコンポーネント間の通信に TCP が適切かどうかを判断するには、ネットワークレイテンシーに注目します。クライアントアプリケーションとサーバー間のネットワークレイテンシーが高い場合、3 方向ハンドシェイクに時間がかかり、アプリケーションの応答性に影響を与える可能性があります。Time to First Byte (TTFB) や Round-Trip Time (RTT) などのメトリクスを使用すると、ネットワークレイテンシーを測定できます。ワークロードがユーザーに動的コンテンツを提供する場合は、 Amazon CloudFrontの使用を検討します。これにより、動的コンテンツの各オリジンへの永続的な接続が確立され、各クライアントリクエストの速度を低下させる接続設定時間が必要なくなります。
3. TCP や UDP で TLS を使用すると、暗号化と復号化の影響により、レイテンシーが増大し、ワークロードのスループットが低下する可能性があります。このようなワークロードの場合、バックエンドインスタンスではなく、 Elastic Load Balancing で SSL/TLS オフロードを行い、ロードバランサーで SSL/TLS 暗号化および復号化プロセスの処理を行ってワークロードのパフォーマンスを向上させることを検討します。これは、バックエンドインスタンスの CPU 使用率の低減、パフォーマンスの向上、キャパシティー増大につながります。
4. 認証と認可、ロギング、DNS、IoT、ストリーミングメディアなどの UDP プロトコルに依存するサービスを展開し、ワークロードのパフォーマンスと信頼性を向上させるには、 Network Load Balancer (NLB) を使用します。NLB は着信 UDP トラフィックを複数のターゲットに分散するため、ワークロードの水平方向のスケーリング、キャパシティの増大、単一のターゲットのオーバーヘッドの低減につながります。
5. ハイパフォーマンスコンピューティング (HPC) のワークロードの場合は、SRD プロトコルを使用する Elastic Network Adapter (ENA) Express 機能を利用することを検討します。この機能は、EC2 インスタンス間のネットワークトラフィックでシングルフロー帯域幅を増大し (25 Gbps)、テールレイテンシーを短縮して (99.9 パーセンタイル)、ネットワークパフォーマンスを向上させます。

6. ワークロードコンポーネント間または gRPC クライアントとサービス間で gRPC (リモートプロシージャコール) トラフィックをルーティングして負荷分散するには、 Application Load Balancer (ALB) を使用します。gRPC は TCP ベースの HTTP/2 プロトコルをトランスポートに使用しており、ネットワークフットプリントの軽量化、圧縮、効率的なバイナリ形式のシリアル化、多言語サポート、双方向ストリーミングなどのパフォーマンス上の利点が得られます。

リソース

関連するドキュメント:

- How to route UDP traffic into Kubernetes
- Application Load Balancer
- Linux での EC2 拡張ネットワーキング
- Windows での EC2 拡張ネットワーキング
- EC2 プレイスメントグループ
- Linux インスタンスで Elastic Network Adapter (ENA) を使用して拡張ネットワークを有効にする
- Network Load Balancer
- AWS ネットワークとコンテンツ配信
- Amazon Route 53 でレイテンシーベースルーティングへ移行する
- VPC エンドポイント

関連動画:

- AWS re:Invent 2022 – Scaling network performance on next-gen Amazon Elastic Compute Cloud instances
- AWS re:Invent 2022 – Application networking foundations

関連する例:

- AWS Transit Gateway and Scalable Security Solutions
- AWS Networking Workshops

PERF04-BP06 ネットワーク要件に基づいてワークロードのロケーションを選択する

ネットワークのレイテンシー短縮、スループット向上、ページの読み込み時間とデータ転送時間の短縮による最適なユーザーエクスペリエンス提供に向けて、リソースプレイスメントのオプションを評価します。

一般的なアンチパターン:

- すべてのワークロードリソースを 1 つの地理的場所に統合する。
- ワークロードのエンドユーザーではなく、自分の所在地に最も近いリージョンを選んでいる。

このベストプラクティスを確立するメリット: ユーザーエクスペリエンスは、ユーザーとアプリケーションの間のレイテンシーに大きく影響されます。適切な AWS リージョン と AWS のプライベートなグローバルネットワークを使用することで、レイテンシ–を減らし、リモートユーザーにより良いエクスペリエンスを提供できます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

Amazon EC2 インスタンスなどのリソースは、AWS リージョン、AWS Local Zones、AWS Outposts、または AWS Wavelength のゾーン内のアベイラビリティーゾーンに配置します。この口ケーション選択が、特定のユーザーの口ケーションからのネットワークレイテンシーとスループットに影響を及ぼします。Amazon CloudFront と AWS Global Accelerator などのエッジサービスを利用して、エッジロケーションにコンテンツをキャッシュするか、AWS グローバルネットワークを介してユーザーにワークロードへの最適なパスを提供することにより、ネットワークパフォーマンスを改善することもできます。

Amazon EC2 にはネットワーク用のプレイスメントグループが用意されています。プレイスメントグループは、レイテンシーを減らすためにインスタンスを論理的にグループ化したものです。サポートされているインスタンスタイプを使ったプレイスメントグループと Elastic Network Adapter (ENA) を使用することにより、ワークロードを低レイテンシー、低ジッターの 25 Gbps ネットワークに参加させることができます。プレイスメントグループは、低ネットワークレイテンシー、高ネットワークスループット、またはその両方からメリットを得るワークロードに推奨されます。

レイテンシーの影響を受けやすいサービスは、Amazon CloudFront などの AWS グローバルネットワークを使用し、エッジロケーションで提供します。これらのエッジロケーションは一般に、コンテ

ンツ配信ネットワーク (CDN) およびドメインネームシステム (DNS) などのサービスを提供します。これらのサービスをエッジで使用することにより、ワークロードがコンテンツまたは DNS 解決のリクエストに低いレイテンシーで応答できるようになります。これらのサービスは、コンテンツのジオターゲティング (エンドユーザーの位置に基づいて異なるコンテンツを提供) などの地理的なサービス、またはエンドユーザーを最寄りのリージョンに誘導するレイテンシーベースルーティング (最小レイテンシー) も提供します。

エッジサービスを使用してレイテンシーを低減し、コンテンツキャッシングを有効化します。これらのアプローチから最大限のメリットを得るために、DNS と HTTP/HTTPS の両方でキャッシュ制御を正しく設定してください。

実装手順

- VPC のネットワークインターフェイスとの間で行き来する IP トラフィックに関する情報を把握します。
  - VPC フローログを使用した IP トラフィックのログ記録
  - How the client IP address is preserved in AWS Global Accelerator
- ワークロードのネットワークアクセスパターンを分析して、ユーザーがアプリケーションをどのように使用しているかを特定します。
  - Amazon CloudWatch や AWS CloudTrail などのモニタリングツールを使用して、ネットワークアクティビティに関するデータを収集します。
  - データを分析して、ネットワークアクセスパターンを特定します。
- 以下の主な要素に基づいて、ワークロードのデプロイに適切なリージョンを選択します。
  - データの場所: 大量のデータを使用するアプリケーション (ビッグデータや機械学習など) では、アプリケーションコードをできるだけデータの近くで実行してください。
  - ユーザーの場所: ユーザー向けアプリケーションの場合は、ワークロードのユーザーに近いリージョン (または複数のリージョン) を選択します。
  - その他の制約: 「What to Consider when Selecting a Region for your Workloads」で説明されているように、コストやコンプライアンスなどの制約を考慮します。
- 動画レンダリングなどのワークロードを実行するには、AWS Local Zones を使用します。ローカルゾーンでは、エンドユーザーの近くにコンピューティングリソースやストレージリソースがあるというメリットが得られます。
- オンプレミスに置いておく必要があるが、AWS にある他のワークロードとシームレスに連携させたいワークロードでは、AWS Outposts を使用します。

- 高解像度ライブビデオストリーミング、Hi-Fi 音源、拡張現実および仮想現実 (AR/VR) などのアプリケーションでは、5G デバイス向けに超低レイテンシーが必要です。このようなアプリケーションについては、AWS Wavelength を検討します。AWS Wavelength は AWS のコンピューティングおよびストレージサービスを 5G ネットワーク内に組み込み、超低レイテンシーアプリケーションの開発、デプロイ、スケーリングのためのモバイルエッジコンピューティングインフラストラクチャを提供します。
- ローカルキャッシュまたは AWS キャッシュソリューションを、頻繁に使用するアセットに使用すると、パフォーマンスを向上させ、データ移動を削減し、環境への影響を低減できます。

Service	When to use
Amazon CloudFront	画像、スクリプト、動画などの静的コンテンツだけでなく、API 応答やウェブアプリケーションなどの動的コンテンツのキャッシュに使用します。
Amazon ElastiCache	ウェブアプリケーションのコンテンツをキャッシュします。
DynamoDB Accelerator	DynamoDB テーブルにインメモリアクセラレーションを追加します。

- 以下のように、ワークロードのユーザーの近くでコードを実行できるサービスを使用します。

Service	When to use
Lambda@edge	オブジェクトがキャッシュにないときに開始される、コンピューティング負荷の高いオペレーションに使用します。
Amazon CloudFront 関数	HTTP リクエストまたはレスポンス操作など、短時間実行の関数で実行できるシンプルなユースケースに使用します。
AWS IoT Greengrass	接続されたデバイスのローカルコンピューティング、メッセージング、データキャッシュを実行します。

- アプリケーションによっては、最初のバイトのレイテンシーとジッターを低減してスループットを向上することによる、固定エントリポイントまたはより高いパフォーマンスが必要となります。このようなアプリケーションは、エニーキャスト IP アドレスとエッジロケーションで TCP ターミネーションを提供するネットワークサービスの利点を活用できます。AWS Global Accelerator を使用すると、アプリケーションのパフォーマンスを最大 60% 向上し、マルチリージョンアーキテクチャの迅速なフェイルオーバーを実現します。AWS Global Accelerator を使用すると、1 つまたは複数の AWS リージョン でホストされるアプリケーションの固定エントリポイントとして機能する静的エニーキャスト IP アドレスが利用できます。このような IP アドレスを使用すると、トラフィックはユーザーのできるだけ近くの AWS グローバルネットワークに入ることができます。AWS Global Accelerator は、クライアントとクライアントに最も近い AWS エッジロケーションの間で TCP 接続を確立することにより、初期接続設定時間を短縮します。TCP/UDP ワークロードのパフォーマンス向上、マルチリージョンアーキテクチャの迅速なフェイルオーバーを実現するには、AWS Global Accelerator の使用を検討します。

リソース

関連するベストプラクティス:

- COST07-BP02 コストに基づいてリージョンを選択する
- COST08-BP03 データ転送コストを削減するサービスを実装する
- REL10-BP01 複数の場所にワークロードをデプロイする
- REL10-BP02 マルチロケーションデプロイ用の適切な場所の選択
- SUS01-BP01 ビジネス要件と持続可能性の目標の両方に基づいてリージョンを選択する
- SUS02-BP04 ネットワーク要件に基づいてワークロードの地理的配置を最適化する
- SUS04-BP07 ネットワーク間でのデータ移動を最小限に抑える

関連するドキュメント:

- AWS グローバルインフラストラクチャ
- AWS Local Zones and AWS Outposts, choosing the right technology for your edge workload (AWS Local Zones と AWS Outpost、エッジのワークロードに適したテクノロジーの選択)
- プレイスメントグループ
- AWS Local Zones
- AWS Outposts

- AWS Wavelength
- Amazon CloudFront
- AWS Global Accelerator
- AWS Direct Connect
- AWS Site-to-Site VPN
- Amazon Route 53

関連動画:

- AWS Local Zones Explainer Video (AWS Local Zones 説明動画)
- AWS Outposts: Overview and How it Works
- AWS re:Invent 2023 - A migration strategy for edge and on-premises workloads
- AWS re:Invent 2021 - AWS Outposts: Bringing the AWS experience on premises (AWS re:Invent 2021 - AWS Outposts: AWS エクスペリエンスをオンプレミスで実現)
- AWS re:Invent 2020: AWS Wavelength: Run apps with ultra-low latency at 5G edge (AWS re:Invent 2020: AWS Wavelength: 5G エッジで超低レイテンシーでアプリケーションを実行する)
- AWS re:Invent 2022 - AWS Local Zones: Building applications for a distributed edge (AWS re:Invent 2022 - AWS Local Zones: 分散エッジ向けアプリケーションの構築)
- AWS re:Invent 2021 - Building low-latency websites with Amazon CloudFront (AWS re:Invent 2021 - Amazon CloudFront を使用した低レイテンシーのウェブサイトの構築)
- AWS re:Invent 2022 - Improve performance and availability with AWS Global Accelerator(AWS re:Invent 2022 - AWS Global Accelerator を使用してパフォーマンスと可用性を向上する)
- AWS re:Invent 2022 - Build your global wide area network using AWS(AWS re:Invent 2022 - AWS を使用してグローバルなワイドエリアネットワークを構築)
- AWS re:Invent 2020: Global traffic management with Amazon Route 53 (AWS re:Invent 2020: Amazon Route 53 を使用したグローバルトラフィック管理)

関連する例:

- AWS Global Accelerator Custom Routing Workshop
- Handling Rewrites and Redirects using Edge Functions (Edge 関数を使用した書き換えとリダイレクトの処理)

PERF04-BP07 メトリクスに基づいてネットワーク設定を最適化する

収集して分析したデータを使用して、ネットワーク設定の最適化に関する十分な知識に基づいた意思決定を行います。

一般的なアンチパターン:

- パフォーマンス関連の問題はすべてアプリケーション関連であると想定している。
- ワークロードをデプロイしたロケーションに近いロケーションからのみ、ネットワークパフォーマンスをテストする。
- ネットワークサービスの設定はすべてデフォルトにしている。
- 十分なキャパシティを提供するためにネットワークリソースを過剰プロビジョニングしている。

このベストプラクティスを活用するメリット: AWS ネットワークの必要なメトリクスを収集し、ネットワークモニタリングツールを実装すると、ネットワークパフォーマンスを把握し、ネットワーク設定を最適化できます。

このベストプラクティスを活用しない場合のリスクレベル: 低

実装のガイダンス

AWS ネットワークリソースの利用方法とネットワーク設定を最適化する方法を理解するには、VPC、サブネット、またはネットワークインターフェイス間のトラフィックをモニタリングすることが重要です。以下の AWS ネットワークツールを使用すると、トラフィックの使用状況、ネットワークアクセス状況、ログに関する情報を詳細に調査できます。

実装手順

- レイテンシーやパケットロスなど、収集する主要なパフォーマンスメトリクスを特定します。AWS には、これらのメトリクスの収集に役立ついくつかのツールが用意されています。以下のツールを使用すると、トラフィックの使用状況、ネットワークアクセス状況、ログに関する情報を詳細に調査できます。

AWS ツール	Amazon VPC IP Address Manager を使用する
場所。	IPAM を使用して、AWS とオンプレミスのワークロードの IP アドレスのプランニング、追跡、モニタリングを行います。これは IP アドレスの使用と割り当てを最適化するためのベストプラクティスです。
VPC フローログ	VPC フローログを使用して、VPC 内のネットワークインターフェイス間で送受信される IP トラフィックに関する詳細をキャプチャします。VPC フローログを使用すると、過度に制限的なセキュリティグループルールや過度に寛容なセキュリティグループルールを診断して、ネットワークインターフェイス間のトラフィックの方向を判断できます。
AWS Transit Gateway フローログ	AWS Transit Gateway フローログを使用して、トランジットゲートウェイで送受信される IP トラフィックに関する情報をキャプチャします。
DNS クエリのログ記録	Route 53 が受信するパブリックまたはプライベート DNS クエリに関する情報を口グに記録します。DNS ログを使用すると、要求されたドメインまたはサブドメイン、または DNS クエリに応答した Route 53 エッジロケーションを把握したうえで DNS の設定を最適化できます。

AWS ツール	Amazon VPC IP Address Manager を使用する
Reachability Analyzer	Reachability Analyzer は、ネットワークの到達可能性の解析およびデバッグに役立ちます。Reachability Analyzer は、VPC 内の送信元リソースと宛先リソース間の接続テストを実行できる構成分析ツールです。このツールを使用すると、ネットワーク設定が意図した接続になっているかを確認できます。
Network Access Analyzer	Network Access Analyzer は、リソースへのネットワークアクセスを理解するのに役立ちます。Network Access Analyzer を使用すると、ネットワークのアクセス要件を指定して、指定した要件を満たさない可能性のあるネットワークパスを特定できます。対応するネットワーク設定を最適化すると、ネットワークの状態を理解して検証し、AWS のネットワークがコンプライアンス要件を満たしているかどうかを検証できます。
Amazon CloudWatch	<!--ATMS sidestep. Remove this--> Amazon CloudWatch を使用して、ネットワークオプションの適切なメトリクスをオンにします。ワークロードに適したネットワークメトリクスを選択していることを確認します。例えば、VPC Network Address Usage、VPC NAT Gateway、AWS Transit Gateway、VPN トンネル、AWS Network Firewall、Elastic Load Balancing、AWS Direct Connect などのメトリクスをオンにできます。メトリクスの継続的なモニタリングは、ネットワークの状態と使用状況を観測して理解する優れた方法であり、観測に基づいたネットワーク設定の最適化につながります。

AWS ツール	Amazon VPC IP Address Manager を使用する
AWS Network Manager	AWS Network Manager を使用すると、運用上および計画上の目的で AWS グローバルネットワーク のリアルタイムおよび過去のパフォーマンスをモニタリングできます。Network Manager では、AWS リージョン 間、アベイラビリティーゾーン間、および各アベイラビリティーゾーン内のネットワークレイテンシーを集計できるため、アプリケーションのパフォーマンスと基盤となる AWS ネットワークのパフォーマンスとの関係をよりよく理解できます。
Amazon CloudWatch RUM	Amazon CloudWatch RUM を使用して、ユーザーエクスペリエンスの特定、把握、改善に役立つインサイトが得られるメトリクスを収集します。

- VPC AWS Transit Gateway とフローログを使用して、上位の送信元やアプリケーショントラフィックのパターンを特定します。
- VPC、サブネット、ルーティングなど、現在のネットワークアーキテクチャを評価して最適化します。例として、異なる VPC ピアリングや AWS Transit Gateway がアーキテクチャ内のネットワークの改善にどのように役立つかを評価できます。
- ネットワーク内のルーティングパスを評価して、宛先間の最短パスが常に使用されていることを確認します。その際に Network Access Analyzer が役立ちます。

リソース

関連するドキュメント:

- パブリック DNS クエリのログ記録
- IPAM とは
- Reachability Analyzer とは
- Network Access Analyzer とは
- VPC の CloudWatch メトリクス

- Optimize performance and reduce costs for network analytics with VPC Flow Logs in Apache Parquet format
- Amazon CloudWatch メトリクスを使用してグローバルとコアネットワークをモニタリングする
- Continuously monitor network traffic and resources

関連動画:

- AWS re:Invent 2023 – A developer's guide to cloud networking
- AWS re:Invent 2023 – Ready for what's next? Designing networks for growth and flexibility
- AWS re:Invent 2023 – Advanced VPC designs and new capabilities
- AWS re:Invent 2022 – Dive deep on AWS networking infrastructure
- AWS re:Invent 2020 – Networking best practices and tips with the AWS Well-Architected Framework
- AWS re:Invent 2020 – Monitoring and troubleshooting network traffic

関連する例:

- AWS Networking Workshops
- AWS Network Monitoring
- Observing and diagnosing your network on AWS
- Finding and addressing network misconfigurations on AWS

プロセスと文化

ワークロードを設計する際には、効率的で高性能なクラウドワークロードをより良く実行するために採用できる原則と慣行があります。この重点分野では、クラウドワークロードのパフォーマンス効率を高める文化を取り入れるのに役立つベストプラクティスを紹介します。

この文化を築くために、以下の基本方針を考慮してください。

- コードとしてのインフラストラクチャ: AWS CloudFormation テンプレートなどのアプローチを使用して、インフラストラクチャをコードとして定義します。テンプレートを使用すると、アプリケーションコードと設定のほか、インフラストラクチャをソースコントロールに配置できます。これは、ソフトウェアの開発に使用するものと同じ慣行をインフラストラクチャに適用することを可能にし、イテレーションを迅速に行えるようになります。
- デプロイパイプライン: 継続的インテグレーションと継続的デプロイ (CI/CD) パイプライン (ソースコードのリポジトリ、ビルドシステム、デプロイ、自動化のテストなど) を使用して、インフラストラクチャをデプロイします。これにより、イテレーションの度に、反復可能で、一貫性があり、低コストな方法でデプロイできるようになります。
- 明確に定義されたメトリクス: メトリクスをセットアップしてモニタリングし、主要業績評価指標 (KPI) を把握します。技術メトリクスとビジネスメトリクスの両方を使用することをお勧めします。ウェブサイトまたはモバイルアプリの主なメトリクスは、最初の 1 バイトを受信するまでの時間、または最初のレンダリングまでの時間のキャプチャです。一般的に適用されるその他のメトリクスには、スレッド数、ガベージコレクション率、および待機状態などがあります。リクエストあたりの総累積コストなどのビジネスメトリクスでは、コストを削減する方法をアラートできます。メトリクスを解釈する方法を十分に検討してください。例えば、平均ではなく最大または 99 パーセンタイルを選択できます。
- パフォーマンステストの自動化: デプロイプロセスの一環として、簡単な実行テストが正常に終了した後に、パフォーマンステストを自動的に開始します。自動化により新しい環境が作成され、テストデータなどの初期条件が設定された後に、一連のベンチマークとロードテストが実行されます。これらのテストの結果は、時間の経過に伴うパフォーマンスの変化を追跡できるようにビルドに関連付けてください。テストが長時間実行される場合、パイプラインのこの部分を、他のビルドと同期しないようにできます。または、Amazon EC2 スポットインスタンスを使用して夜間にパフォーマンステストを実行できます。
- ロードの生成: 合成ユーザージャーニーや事前に記録されたユーザージャーニーをレプリケートする一連のテストスクリプトを作成してください。これらは結合スクリプトではなく、べき等スクリプトにしてください。有効な結果を得るために、 プレウォーミング スクリプトを含める必要がある場合もあります。テストスクリプトは、可能な限り本番での使用状況を再現する必要がありま

す。ソフトウェアまたは SaaS (Software-as-a-Service) ソリューションを使用してロードを生成できます。AWS Marketplace ソリューションと スポットインスタンス の使用を検討してください。コスト効率に優れた方法でロードを生成できます。

- パフォーマンスの可視性: 主要なメトリクス、特に各ビルドバージョンに対するメトリクスはチームが確認できるようにする必要があります。これにより、時間の経過に伴う大量の肯定的または否定的な傾向の評価を参照できるようになります。また、エラーや例外でもメトリクスを表示し、確実に作動するシステムをテストするようにしてください。
- 可視化: 可視化テクニックを使用して、パフォーマンスの問題、ホットスポット、待機状態、または使用率が低い状態が発生する場所を明確にします。アーキテクチャ図にパフォーマンスメトリクスを重ね合わせます。コールグラフ、または符号も問題をすばやく特定するために役立ちます。
- 定期的なレビュープロセス: 低パフォーマンスのアーキテクチャは通常、パフォーマンスレビュープロセスが存在しないか、または破綻していることに起因します。アーキテクチャのパフォーマンスレベルが低い場合は、パフォーマンスレビュープロセスを導入することで、反復的な改善を促すことができます。
- 継続的な最適化: クラウドワークロードのパフォーマンス効率を継続的に最適化する文化を取り入れましょう。

ベストプラクティス

- PERF05-BP01 ワークロードの状態とパフォーマンスを測定するための主要業績評価指標 (KPI) を設定する
- PERF05-BP02 モニタリングソリューションを活用して、パフォーマンスが最も重要な分野について把握する
- PERF05-BP03 ワークロードのパフォーマンス向上プロセスを定める
- PERF05-BP04 ワークロードの負荷テストを実施する
- PERF05-BP05 自動化でパフォーマンス関連の問題をプロアクティブに修正する
- PERF05-BP06 ワークロードとサービスを最新の状態に保つ
- PERF05-BP07 メトリクスを定期的に見直す

PERF05-BP01 ワークロードの状態とパフォーマンスを測定するための主要業績評価指標 (KPI) を設定する

ワークロードのパフォーマンスを定量的および定性的に測定する KPI を特定します。KPI は、ビジネス目標に関連するワークロードの健全性とパフォーマンスを測定するのに役立ちます。

一般的なアンチパターン:

- ワークロードについて把握するためだけにシステムレベルのメトリクスをモニタリングし、こうしたメトリクスがビジネスに与える影響を理解していない。
- KPI が標準的なメトリクスデータとして既に発行され、共有されていると思っている。
- 定量的で測定可能な KPI を定義していない。
- KPI をビジネスの目標や戦略とすり合わせていない。

このベストプラクティスを確立するメリット: ワークロードの正常性とパフォーマンスを表す具体的な KPI を特定することで、チームの優先順位をすり合わせ、目指すべきビジネス成果とは何かを定義できます。これらのメトリクスをすべての部門と共有することで、しきい値、期待値、ビジネスへの影響が可視化され、調整を図ることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

KPI を使用すると、ビジネスチームとエンジニアリングチームが、目標と戦略の測定と、こうした要因がどのように組み合わさってビジネス成果を生み出すかについての認識をすり合わせることができます。例えば、ウェブサイトのワークロードには、ページの読み込み時間を全体的なパフォーマンスの指標として使用する場合があります。このメトリクスは、ユーザーエクスペリエンスを測定する複数のデータポイントのうちの 1 つとなります。ページの読み込み時間のしきい値を特定することに加えて、パフォーマンスが満たされない場合に期待される結果やビジネスリスクを文書化する必要があります。ページの読み込み時間が長いと、エンドユーザーに直接影響し、ユーザーエクスペリエンスの評価の低下、ひいては顧客の損失につながる可能性があります。KPI のしきい値を定義するときは、業界のベンチマークとエンドユーザーの期待値の両方を組み合わせます。例えば、現時点での業界のウェブページの読み込みベンチマークが 2 秒以内であっても、エンドユーザーが 1 秒以内での読み込みを期待する場合、KPI を設定する際にこれらのデータポイントの両方を考慮する必要があります。

チームは、リアルタイムの詳細なデータと参照用の履歴データを使用してワークロード KPI を評価し、KPI データにメトリクス計算を実行するダッシュボードを作成して、運用と使用状況に関する洞察を導き出す必要があります。KPI は文書化され、ビジネス目標と戦略をサポートするしきい値を含み、かつモニタリング対象のメトリクスに対応付けられている必要があります。ビジネスの目標および戦略、またはエンドユーザーの要件が変わった場合は、KPI を再検討する必要があります。

実装手順

- 利害関係者を特定する: 開発チームや運用チームなど、主要なビジネス関係者を特定して文書化します。
- 目標を定義する: 特定した利害関係者と協力して、ワークロードの目標を定義し、文書化します。スループット、応答時間、コストなど、ワークロードのパフォーマンス上重要となる側面に加え、ユーザー満足度などのビジネス目標も考慮に入れてください。
- 業界のベストプラクティスを確認する: 業界のベストプラクティスを確認して、ワークロードの目標に沿った関連 KPI を特定します。
- メトリクスを特定する: ワークロード目標に沿ったメトリクスを特定します。パフォーマンスとビジネス目標の測定に役立ちます。これらのメトリクスに基づいて KPI を設定します。メトリクスの例として、平均応答時間や同時ユーザー数などの測定値があります。
- KPI を定義して文書化する: 業界のベストプラクティスとワークロードの目標を使用して、ワークロード KPI のターゲットを設定します。この情報を使用して、重要度またはアラームレベルの KPI しきい値を設定します。KPI が満たされない場合のリスクと影響を特定して文書化します。
- モニタリングを実装する: Amazon CloudWatch や AWS Config などのモニタリングツールを使用して、メトリクスを収集し、KPI を測定します。
- KPI を視覚的に提示する: Amazon QuickSight などのダッシュボードツールを使用して KPI を視覚化し、利害関係者に提示します。
- 分析して最適化する: KPI を定期的に見直して分析し、ワークロードで改善が必要な領域を特定します。利害関係者と協力してこれらの改善を実装します。
- 再検討して改善する: ビジネス目標やワークロードのパフォーマンスに変更があった場合は特に、メトリクスと KPI を定期的に見直してその有効性を評価します。

リソース

関連するドキュメント:

- CloudWatch ドキュメント
- モニタリング、ログ記録、およびパフォーマンス AWS Partner
- AWS observability tools
- The Importance of Key Performance Indicators (KPIs) for Large-Scale Cloud Migrations
- How to track your cost optimization KPIs with the KPI Dashboard
- X-Ray ドキュメント

- Amazon CloudWatch ダッシュボードの使用
- Amazon QuickSight KPI

関連動画:

- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2023 - Manage resource lifecycle events at scale with AWS Health
- AWS re:Invent 2023 - Performance & efficiency at Pinterest: Optimizing the latest instances
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Scaling on AWS for the first 10 million users
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Creating an Effective Metrics Strategy for Your Business | AWS Events

関連する例:

- Creating a dashboard with Amazon QuickSight

PERF05-BP02 モニタリングソリューションを活用して、パフォーマンスが最も重要な分野について把握する

ワークロードのパフォーマンスの向上が効率性やカスタマーエクスペリエンスにプラスの影響を与える分野を理解し、特定します。例えば、カスタマーインタラクションが多いウェブサイトは、エッジサービスを使用してコンテンツ配信をお客様に近い場所へ移動させることでメリットを得ることができます。

一般的なアンチパターン:

- パフォーマンスの問題を検出するには、CPU 使用率やメモリプレッシャーなどの標準的なコンピューティングメトリクスで十分であると考えている。
- 一部のモニタリングソフトウェアで記録されるデフォルトのメトリクスのみを使用している。
- 問題が発生したときにだけメトリクスを確認している。

このベストプラクティスを確立するメリット: パフォーマンスの重要な領域を把握することで、ワークロードの所有者は KPI をモニタリングし、影響の大きい改善点に優先的に取り組むことができます。

このベストプラクティスが確立されていない場合のリスクレベル: 高

実装のガイダンス

エンドツーエンドの追跡を構築して、トラフィックパターン、レイテンシー、重要なパフォーマンス領域を特定します。データアクセスパターンをモニタリングして、低速なクエリや不十分にフラグメント化されパーティション化されたデータを検出します。負荷テストまたはモニタリングを使用して、ワークロードのボトルネックを特定します。

アーキテクチャ、トラフィックパターン、データアクセスパターンを理解し、レイテンシーと処理時間を特定することで、パフォーマンス効率を高めることができます。ワークロードが増加するにつれて、顧客エクスペリエンスに影響を及ぼす可能性のある潜在的なボトルネックを特定できます。この領域を調査したら、デプロイできるソリューションを調査し、パフォーマンスの懸念を取り除きます。

実装手順

- エンドツーエンドのモニタリングを構築して、すべてのワークロードコンポーネントおよびメトリクスをキャプチャします。以下は、AWS のモニタリングソリューションの一部です。

Service	Where to use
Amazon CloudWatch Real-User Monitoring (RUM)	To capture application performance metrics from real user client-side and frontend sessions.
AWS X-Ray	To trace traffic through the application layers and identify latency between components and dependencies. Use X-Ray service maps to see relationships and latency between workload components.
Amazon Relational Database Service Performance Insights	To view database performance metrics and identify performance improvements.

Service	Where to use
Amazon RDS Enhanced Monitoring	To view database OS performance metrics.
Amazon DevOps Guru	To detect abnormal operating patterns so you can identify operational issues before they impact your customers.

- テストを実行してメトリクスを生成し、トラフィックパターン、ボトルネック、および重要なパフォーマンス領域を特定します。テストの実行方法の例として、次のようなものがあります。
  - CloudWatchSynthetic Canaries をセットアップして、Linux の cron ジョブまたはレート式を使用してブラウザベースのユーザーアクティビティをプログラムで模倣するように設定し、長期にわたって一貫したメトリクスを生成します。
  - AWS での分散負荷テストソリューションを使用して、ピークトラフィックを生成するか、予想される増加率でワークロードをテストします。
- メトリクスとテレメトリを評価して、重要なパフォーマンス領域を特定します。これらの領域をチームと一緒にレビューして、モニタリングおよびボトルネックを防ぐためのソリューションについて話し合います。
- パフォーマンスの改善をテストし、データを使用してこれらの変更を計測します。例として、CloudWatch Evidentlyを使用して、ワークロードに対する新しい改善点やパフォーマンスへの影響をテストできます。

リソース

関連するドキュメント:

- What's new in AWS Observability at re:Invent 2023
- Amazon Builders' Library
- X-Ray ドキュメント
- Amazon CloudWatch RUM
- Amazon DevOps Guru

関連動画:

- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability

- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - The Amazon Builders' Library: 25 years of Amazon operational excellence
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- Visual Monitoring of Applications with Amazon CloudWatch Synthetics

関連する例:

- Measure page load time with Amazon CloudWatch Synthetics
- Amazon CloudWatch RUM Web Client
- X-Ray SDK for Python
- AWS での分散負荷テスト

PERF05-BP03 ワークロードのパフォーマンス向上プロセスを定める

新しいサービス、設計パターン、リソースの種類、設定が利用できるようになった時点で、これらを評価するプロセスを明確に定めます。たとえば、新しいインスタンス製品で既存のパフォーマンステストを実行して、ワークロードを向上させる可能性を判断します。

一般的なアンチパターン:

- 現在のアーキテクチャが静的であり、今後更新されないと考えている。
- メトリクスに基づく理由なしで、時間の経過とともにアーキテクチャの変更を導入する。

このベストプラクティスを確立するメリット: アーキテクチャを変更するためのプロセスを定義することで、収集したデータを使用して長期的なワークロード設計に影響を与えることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

ワークロードのパフォーマンスには重要な制約がいくつかあります。その制約を文書化すれば、どのような種類のイノベーションがワークロードのパフォーマンス向上につながるかを把握できます。新

しいサービスやテクノロジーが利用できるようになった場合、この情報を利用して、制約やボトルネックを軽減する方法を見つけます。

ワークロードの重要なパフォーマンス上の制約を特定します。どのようなイノベーションがワークロードパフォーマンスの向上につながるかを知ることができるように、ワークロードのパフォーマンスの制約を文書化します。

実装手順

- KPI を特定する:「PERF05-BP01 ワークロードの状態とパフォーマンスを測定するための主要業績評価指標 (KPI) を設定する」で説明されているように、ワークロードのパフォーマンス KPI を特定して、ワークロードのベースラインを定めます。
- モニタリングを実装する: AWS オブザーバビリティツールを使用してパフォーマンスメトリクスを収集し、KPI を測定します。
- 分析を実施する: 詳細な分析を実施して、「PERF05-BP02 モニタリングソリューションを活用して、パフォーマンスが最も重要な分野について把握する」で説明されているように、ワークロードの中でパフォーマンスが低い領域 (設定やアプリケーションコードなど) を特定します。分析ツールとパフォーマンスツールを使用して、パフォーマンス改善戦略を特定します。
- 改善を検証する: サンドボックス環境または本番前環境を使用して、改善戦略の有効性を検証します。
- 変更を実装する: 変更を本番環境に実装し、ワークロードのパフォーマンスを継続的にモニタリングします。改善点を文書化し、利害関係者に変更点を報告します。
- 再検討して改良する: パフォーマンス改善プロセスを定期的に見直し、強化できる分野を特定します。

リソース

関連するドキュメント:

- AWS ブログ
- AWS の最新情報
- AWS Skill Builder

関連動画:

- AWS re:Invent 2022 - Delivering sustainable, high-performing architectures

- AWS re:Invent 2023 - Optimize cost and performance and track progress toward mitigation
- AWS re:Invent 2022 - AWS optimization: Actionable steps for immediate results
- AWS re:Invent 2022 - Optimize your AWS workloads with best-practice guidance

関連する例:

- AWS Github

PERF05-BP04 ワークロードの負荷テストを実施する

ワークロードの負荷テストを実施して、本番環境の負荷に対応できることを確認し、パフォーマンスのボトルネックを特定します。

一般的なアンチパターン:

- あなたは、ワークロード全体ではなく、ワークロードの個々の部分について負荷テストを行います。
- あなたは、本番環境とは異なるインフラストラクチャで負荷テストを行います。
- あなたは、今後問題が発生する可能性を予測するのに役立てるため、予想される負荷に対してのみ、負荷テストを実施し、それを超える負荷に対しては負荷テストを実施しません。
- Amazon EC2 のネットワーク負荷テストを参照したり、シミュレーションイベントサブミッションフォームを送信することなく負荷テストを実行しています。これは、サービス妨害イベントとみなされ、テストの実行の失敗につながります。

このベストプラクティスを確立するメリット: 負荷テストでパフォーマンスを測定すると、負荷が増加したときにどの部分が影響を受けるかがわかります。これにより、必要な変更がワークロードに影響を与える前に予測できるようになります。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

クラウドでの負荷テストは、予想されるユーザー負荷を考慮して、現実的な条件下でクラウドワークロードのパフォーマンスを測定するプロセスです。このプロセスでは、本番環境と同様のクラウド環境をプロビジョニングし、負荷テストツールを使用して負荷を生成したうえで、メトリクスを分析してワークロードが現実的な負荷に対応できるかを評価します。ロードテストは、本番データの合成バージョンまたはサニタイズバージョン (機密情報または識別情報を削除) を使用して実行する必要

があります。デリバリーパイプラインの一環として負荷テストを自動的に実行し、その結果を事前定義された KPI およびしきい値と比較します。このプロセスにより、必要なパフォーマンスを継続的に達成できます。

実装手順

- テスト目標を定義する: スループットや応答時間など、ワークロードで評価対象とするパフォーマンスの側面を特定します。
- テストツールを選択する: ワークロードに適した負荷テストツールを選択して設定します。
- 環境を設定する: 本番環境に基づいてテスト環境を設定します。AWS のサービスを使用して、アーキテクチャをテストするための本番規模の環境を実行することができます。
- モニタリングを実装する: Amazon CloudWatch などのモニタリングツールを使用して、アーキテクチャ内のリソース全体のメトリクスを収集します。カスタムメトリクスを収集して発行することもできます。
- シナリオを定義する: 負荷テストのシナリオとパラメータ (テスト期間やユーザー数など) を定義します。
- 負荷テストを実施する: テストシナリオを大規模に実施します。AWS クラウドを活用してワークロードをテストし、どこでスケールしないのか、あるいは非線形にスケールしているのかを発見してください例えば、低コストで負荷を生成し、本番前にボトルネックを発見するには、スポットインスタンスを使用します。
- テスト結果を分析する: 結果を分析して、パフォーマンスのボトルネックおよび改善が必要な領域を特定します。
- 調査結果を文書化して共有する: 調査結果と推奨事項を文書化して報告します。この情報を利害関係者と共有して、パフォーマンスの最適化戦略に関して十分な情報に基づいた意思決定を行えるようにします。
- 継続的に繰り返す: 負荷テストは定期的に実行してください。特に、システム変更や更新を行った後は実行する必要があります。

リソース

関連するドキュメント:

- Amazon CloudWatch RUM
- Amazon CloudWatch Synthetics
- AWS での分散負荷テスト

関連動画:

- AWS Summit ANZ 2023: Accelerate with confidence through AWS Distributed Load Testing
- AWS re:Invent 2022 - Scaling on AWS for your first 10 million users
- Solving with AWS Solutions: Distributed Load Testing

- AWS re:Invent 2021 - Optimize applications through end user insights with Amazon CloudWatch RUM (AWS re:Invent 2021 - Amazon CloudWatch RUM を使用してエンドユーザーインサイトを介しアプリケーションを最適化する)
- Amazon CloudWatch Synthetics のデモ

関連する例:

- AWS での分散負荷テスト

PERF05-BP05 自動化でパフォーマンス関連の問題をプロアクティブに修正する

主要業績評価指標 (KPI) をモニタリングおよびアラート発行システムと組み合わせて使用し、パフォーマンス関連の問題に積極的に対処します。

一般的なアンチパターン:

- 運用スタッフのみに対して、ワークロードに運用上の変更を加えることを許可する。
- プロアクティブな修復を行うことなく、すべてのアラームが運用チームに届くようにしている。

このベストプラクティスを確立するメリット: アラームアクションをプロアクティブに修正することで、サポートスタッフは自動的に対応できない項目に集中できます。これにより、運用スタッフがすべてのアラームの対応に忙殺されることがなくなり、代わりに重要なアラームのみに集中できます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

アラームを使用して、可能な場合には自動的に問題を修正するアクションを呼び出します。自動化された対応が不可能な場合は、対応できるシステムにアラームをエスカレートします。例えば、期待

される主要業績評価指標 (KPI) 値を予測し、それらが特定のしきい値を超えた場合にアラームを発行できるシステム、または KPI が期待される値の範囲外である場合に、デプロイメントを自動的に停止、またはロールバックできるツールなどが考えられます。

実行中のワークロードのパフォーマンスを目で見て確認できるようにするプロセスを実装します。モニタリングダッシュボードを構築し、パフォーマンス期待のベースラインとなる基準を確立して、ワークロードが最適に機能しているかどうかを判断します。

実装手順

- 修正ワークフローの特定: 自動的に修正できるパフォーマンスの問題を特定して理解します。問題の根本原因をよりよく理解するために、Amazon CloudWatch や AWS X-Ray など AWS のモニタリングソリューションを使用します。
- オートメーションプロセスの定義: 問題の自動修正に使用できる順を追った修正プロセスを作成します。
- 開始イベントの設定: 修正プロセスを自動的に開始するようにイベントを設定します。例えば、CPU 使用率が特定のしきい値に達したときにインスタンスを自動的に再起動するトリガーを定義できます。
- 修正の自動化: AWS のサービスとテクノロジーを使用して修正プロセスを自動化します。例えば、AWS Systems Manager Automation を使用すると、安全かつスケーラブルに修正プロセスを自動化できます。問題がうまく解決されない場合は、必ず自己修復ロジックを使用して変更を元に戻してください。
- ワークフローのテスト: 自動修正プロセスを本番前環境でテストします。
- ワークフローの実装: 自動修正を本番環境に実装します。
- プレイブックの作成: 開始イベント、修正ロジック、実行されたアクションなど、修正計画の手順を簡単に記したプレイブックを作成して文書化します。自動修正イベントに適切に対応できるように、必ず関係者へのトレーニングを行ってください。
- 見直しと改良: 自動修正ワークフローの有効性を定期的に評価します。必要に応じて開始イベントと修正ロジックを調整します。

リソース

関連するドキュメント:

- CloudWatch ドキュメント
- モニタリング、ログ記録、およびパフォーマンス AWS Partner Network パートナー

- X-Ray ドキュメント
- Using Alarms and Alarm Actions in CloudWatch
- Build a Cloud Automation Practice for Operational Excellence: Best Practices from AWS Managed Services
- Automate your Amazon Redshift performance tuning with automatic table optimization

関連動画:

- AWS re:Invent 2023 - Strategies for automated scaling, remediation, and smart self-healing
- AWS re:Invent 2023 - [LAUNCH] Application monitoring for modern workloads
- AWS re:Invent 2023 - Implementing application observability
- AWS re:Invent 2021 - Intelligently automating cloud operations
- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - Automating patch management and compliance using AWS
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS
- AWS re:Invent 2021 -{New Launch} Automatically detect and resolve issues with Amazon DevOps Guru
- AWS re:Invent 2023 - Centralize your operations

関連する例:

- CloudWatch Logs Customize Alarms

PERF05-BP06 ワークロードとサービスを最新の状態に保つ

新しいクラウドサービスと機能の最新情報を入手し、効率的な機能を取り入れ、問題を取り除き、ワークロードの全体的なパフォーマンス効率を向上させます。

一般的なアンチパターン:

- 現在のアーキテクチャが今後は静的なものとなり、しばらく更新されないと考えている。
- 更新されたソフトウェアおよびパッケージがワークロードと互換性があるかどうかを評価するためのシステムまたは定期的な予定がない。

このベストプラクティスを確立するメリット: 新しいサービスやオファリングに関する最新情報を入手するプロセスを確立することで、新しい機能を取り入れ、問題を解決し、ワークロードパフォーマンスを向上させることができます。

このベストプラクティスが確立されていない場合のリスクレベル: 低

実装のガイダンス

新しいサービス、設計パターン、製品が利用可能になったら、パフォーマンスを向上させる方法を検討します。評価、社内でのディスカッション、または外部分析を通じて、これらのリリースとサービスのどれがワークロードのパフォーマンスまたは効率性を向上させるかを判断します。ワークロードに関連するアップデート、新しい機能、サービスを評価するプロセスを定義します。例えば、新テクノロジーを使用する PoC (概念実証) の構築や内部グループとの協議などのプロセスが考えられます。新しいアイデアやサービスを試す場合、パフォーマンステストを実施して、ワークロードのパフォーマンスへの影響を測定します。

実装手順

- ワークロードのインベントリを作成する: ワークロードソフトウェアおよびアーキテクチャのインベントリを作成して、更新する必要があるコンポーネントを特定します。
- 更新のソースを特定する: ワークロードコンポーネントに関連するニュースと更新のソースを特定します。例えば、AWS の最新情報ブログを購読して、ワークロードのコンポーネントに合った製品を確認できます。RSS フィードを購読するか、メール購読を管理できます。
- 更新スケジュールを定義する: ワークロード用の新しいサービスと機能を評価するためのスケジュールを設定します。
  - AWS Systems Manager インベントリを使用して、Amazon EC2 インスタンスからオペレーティングシステム (OS)、アプリケーション、インスタンスのメタデータを収集し、どのインスタンスがソフトウェアポリシーで要求されるソフトウェアと設定を実行しているか、どのインスタンスが更新する必要があるかを迅速に把握できます。
- 新しい更新を評価する: ワークロードのコンポーネントを更新する方法を理解します。クラウドの俊敏性を利用して、新しい機能によってワークロードがどのように改善するかをすばやくテストし、パフォーマンス効率を向上させます。
- オートメーションを使用する: 更新プロセスにオートメーションを使用して、新しい機能のデプロイに要する作業を軽減し、手動プロセスに起因するエラーを抑制します。
  - CI/CD を使用して、AMI、コンテナイメージなど、クラウドアプリケーションに関連するアーティファクトを自動的に更新できます。

  - AWS Systems Manager Patch Manager などのツールを使用してシステム更新のプロセスを自動化し、AWS Systems Manager Maintenance Windows でを使用してアクティビティをスケジュールできます。
- プロセスを文書化する: アップデートと新しいサービスの評価プロセスを文書化します。アップデートや新しいサービスを調査、テスト、実験、検証するために必要な時間と場所を所有者に提供します。文書化したビジネスの要件と KPI を参照して、どのアップデートがビジネスにメリットをもたらすかの優先順位を付けます。

リソース

関連するドキュメント:

- AWS ブログ
- AWS の最新情報
- Implementing up-to-date images with automated EC2 Image Builder pipelines

関連動画:

- AWS re:Inforce 2022 - Automating patch management and compliance using AWS
- All Things Patch: AWS Systems Manager | AWS Events

関連する例:

- Inventory and Patch Management
- 1 つの可観測性ワークショップ

PERF05-BP07 メトリクスを定期的に見直す

定期的なメンテナンスの一環として、またはイベントやインシデントに応じて、収集対象のメトリクスを見直します。この見直しを通じて、どのメトリクスが問題対応の鍵となったか、またどのメトリクスを追加で追跡すると問題の特定、対応、防止に役立つと思われるかを特定します。

一般的なアンチパターン:

- メトリクスを長期間アラーム状態のままにする。
- 自動システムによって実行できないアラームを作成する。

このベストプラクティスを確立するメリット: 収集されている指標を継続的に見直して、問題が適切に特定、対処、防止されているかどうかを確認できます。また、メトリクスは、長期間アラーム状態のままとなった場合にも、陳腐化することがあります。

このベストプラクティスが確立されていない場合のリスクレベル: 中

実装のガイダンス

メトリクスの収集とモニタリングを継続的に改善します。インシデントやイベントへの対応の一環として、問題解決に役立ったメトリクスと、問題解決に役立った可能性があるものの、現在は追跡されていないメトリクスを評価します。この方法を使用して収集するメトリクスの品質を高め、今後のインシデントを防止、またはより迅速に解決できるようにします。

インシデントやイベントへの対応の一環として、問題解決に役立ったメトリクスと、問題解決に役立った可能性があるものの、現在は追跡されていないメトリクスを評価します。これを使用して、収集するメトリクスの品質を高め、今後のインシデントを防止、またはより迅速に解決できるようにします。

実装手順

- メトリクスを定義する: モニタリングを行う重要なパフォーマンスメトリクスを定義します。応答時間やリソース使用率など、ワークロード目標に沿ったメトリクスを定義します。
- ベースラインを設定する: 各メトリクスのベースラインと目標値を設定します。ベースラインの設定により、逸脱や異常を特定するための基準点が明確になります。
- 頻度を設定する: 重要なメトリクスをレビューする頻度 (毎週、毎月など) を設定します。
- パフォーマンス上の問題を特定する: 各レビューでは、傾向やベースライン値からの逸脱を評価します。パフォーマンスのボトルネックや異常がないか調べます。特定された問題については、詳細な根本原因分析を実施して、問題の背後にある主な理由を把握します。
- 是正措置を特定する: 分析結果に基づいて是正措置を特定します。これには、パラメータの調整、バグの修正、リソースのスケーリングが含まれます。
- 調査結果を文書化する: 特定された問題、根本原因、是正措置など、調査結果を文書化します。
- 反復して改善する: メトリクスのレビュープロセスを継続的に評価し、改善します。前回のレビューで学んだ教訓を活かして、徐々にプロセスを強化します。

リソース

関連するドキュメント:

- CloudWatch ドキュメント
- CloudWatch エージェントを使用して Amazon EC2 インスタンスとオンプレミスサーバーからメトリクスとログを収集する
- Query your metrics with CloudWatch Metrics Insights
- モニタリング、ログ記録、およびパフォーマンス AWS Partner Network パートナー
- X-Ray ドキュメント

関連動画:

- AWS re:Invent 2022 - Setting up controls at scale in your AWS environment
- AWS re:Invent 2022 - How Amazon uses better metrics for improved website performance
- AWS re:Invent 2023 - Building an effective observability strategy
- AWS Summit SF 2022 - Full-stack observability and application monitoring with AWS
- AWS re:Invent 2023 - Take a load off: Diagnose & resolve performance issues with Amazon RDS

関連する例:

- Creating a dashboard with Amazon QuickSight
- CloudWatch Dashboards

まとめ

パフォーマンス効率を達成して維持するにはデータ駆動型のアプローチが必要になります。高パフォーマンスを得るために最適化できるアクセスパターンとトレードオフについて積極的に検討してください。ベンチマークと負荷テストに基づくレビュープロセスを使用することで、適切なリソースタイプと設定を選択できるようになります。インフラストラクチャをコードとして扱うことにより、アーキテクチャを迅速かつ安全に進化させながら、データを使用してアーキテクチャに関する事実に基づいた意思決定を行うことができます。アクティブ/パッシブモニタリングの組み合わせの導入は、アーキテクチャのパフォーマンスが時間と共に劣化しないことを確実にします。

AWS は、ビジネス価値を提供しながら、効率的に機能するアーキテクチャの構築を支援するべく尽力しています。本書で説明したツールと技法を使用することで、成功を確かなものにしてください。

寄稿者

本ドキュメントは、次の人物および組織が寄稿しました。

- Amazon Web Services、Senior Efficiency Lead Solutions Architect、Sam Mokhtari
- Amazon Web Services、ソリューションアーキテクト、Josh Hart
- Amazon Web Services、ソリューションアーキテクト、Richard Trabing
- Amazon Web Services、プリンシパルソリューションアーキテクト、Brett Looney
- Amazon Web Services、プリンシパルソリューションアーキテクト、Nina Vogl
- Amazon Web Services、ソリューションアーキテクト、Eric Pullen
- Amazon Web Services、スペシャリスト SA マネージャー、Julien Lépine
- Amazon Web Services、ソリューションアーキテクト、Ronnen Slasky

その他の資料

追加のサポートについては、以下の資料を参照してください。

- AWS Well-Architected Framework
- AWS アーキテクチャセンター

改訂履歴

このホワイトペーパーの更新に関する通知を受け取るには、RSS フィードをサブスクライブしてください。

変更	説明	日付
ホワイトペーパーの更新	新しい実装ガイダンスを使用してベストプラクティスを更新。	June 27, 2024
メジャーな更新と再構成	柱を 5 つのベストプラクティス領域に再構成 (8 分野から減少)。コンテンツを 5 つの領域に統合し、更新。 新しいベストプラクティス領域は アーキテクチャの選択、コンピューティングとハードウェア、データ管理、ネットワークとコンテンツ配信、 プロセスと文化。	October 3, 2023
マイナーな更新	インクルーシブでない表現を削除。	April 13, 2023
新しいフレームワークの更新	規範ガイダンスを使用してベストプラクティスを更新、および新しいベストプラクティスを追加。	April 10, 2023
ホワイトペーパーの更新	新しい実装ガイダンスを使用してベストプラクティスを更新。	December 15, 2022
ホワイトペーパーの更新	ベストプラクティスに加筆し、改善計画を追加。	October 20, 2022

マイナーな更新	インクルーシブでない表現を削除しました。	April 22, 2022
マイナーな更新	イントロダクションに持続可能性の柱を追加。	December 2, 2021
マイナーな更新	リンクを更新しました。	March 10, 2021
マイナーな更新	AWS Lambda タイムアウトを 900 秒に変更し、Amazon Keyspaces (for Apache Cassandra) の名前を修正。	October 5, 2020
マイナーな更新	壊れたリンクを修正。	July 15, 2020
新しいフレームワークの更新	コンテンツにおける重要な改訂と更新	July 8, 2020
ホワイトペーパーの更新	文法上の問題に関する小規模更新	July 1, 2018
ホワイトペーパーの更新	AWS における変更を反映するためにホワイトペーパーを更新しました。	November 1, 2017
初版発行	パフォーマンス効率の柱 - AWS Well-Architected フレームワークを公開しました。	November 1, 2016

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the AWS の用語集 Reference.