AWS ホワイトペーパー

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

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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 ホワイトペーパー

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスは、Amazon のものではない製品またはサービスと関連付けてはならず、また、お客様に混乱を招くような形や Amazon の信用を傷つけたり失わせたりする形で使用することはできません。Amazon が所有しない商標はすべてそれぞれの所有者に所属します。所有者は必ずしも Amazon と提携していたり、関連しているわけではありません。また、Amazon 後援を受けているとはかぎりません。

Table of Contents

要約 1
要約 1
はじめに 2
Amazon S3 のパフォーマンスのガイドライン 4
パフォーマンスを測定する 4
ストレージ接続を水平にスケールする 4
バイト範囲のフェッチを使用する 5
レイテンシーの影響を受けやすいアプリケーションのリクエストを再試行する 5
同じ AWS リージョンで Amazon S3 (ストレージ) と Amazon EC2 (コンピューティング) を組み合わせる 5
Amazon S3 Transfer Acceleration を使用して距離によるレイテンシーを最小限に抑える 6
最新バージョンの AWS SDK を使用する 6
Amazon S3 のパフォーマンス設計パターン 7
頻繁にアクセスされるコンテンツにキャッシュを使用する 7
レイテンシーの影響を受けやすいアプリケーションのタイムアウトと再試行 8
高スループットのための水平スケーリングとリクエスト並列化 9
Amazon S3 Transfer Acceleration を使用して長距離間のデータ転送を高速化する 10
寄稿者 12
改訂履歴 13
注意 14

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

初版発行日: 2019 年 6 月 (改訂履歴)

要約

Amazon S3 に対してストレージのアップロードや取得を行うアプリケーションを作成する場合は、AWS のベストプラクティスガイドラインに従ってパフォーマンスを最適化してください。AWS では、より詳細なパフォーマンス設計パターンも提供しています。

はじめに

Amazon S3 のストレージに対してアップロードおよび取得を行う際に、アプリケーションはリクエストのパフォーマンスとして 1 秒あたり数千のトランザクションを容易に達成できます。Amazon S3 は、高いリクエストレートに自動的にスケールされます。例えば、アプリケーションでバケット内のプレフィックスごとに 1 秒あたり 3,500 回以上の PUT/COPY/POST/DELETE リクエストと 5,500 回以上の GET/HEAD リクエストを達成できます。バケット内のプレフィックスの数に制限はありません。読み取りを並列化することによって読み取りまたは書き込みのパフォーマンスを向上させることができます。たとえば、Amazon S3 バケットに 10 個のプレフィックスを作成して読み取りを並列化すると、読み取りパフォーマンスを 1 秒あたり 55,000 回の読み取りリクエストにスケールできます。

Amazon S3 上のデータレイクアプリケーションによっては、ペタバイトを超えるデータに対して実行されるクエリで数百万から数十億のオブジェクトをスキャンします。これらのデータレイクアプリケーションは、Amazon EC2 インスタンスのネットワークインターフェイスの使用を最大限に高めて、単一のインスタンスで最大 100 Gb/秒の転送レートを実現しています。その後、これらのアプリケーションは、複数のインスタンスにわたってスループットを集約して 1 秒あたり複数テラバイトを確保します。

ソーシャルメディアメッセージングアプリケーションなどの他のアプリケーションは、レイテンシーの影響を受けやすいアプリケーションです。このようなアプリケーションでは、小さなオブジェクト (大きなオブジェクトの場合は最初のバイトを受け取るまで) で約 100～200 ミリ秒の一定のレイテンシーを実現できます。

他の AWS サービスもさまざまなアプリケーションアーキテクチャのパフォーマンスの高速化に役立ちます。例えば、HTTP 接続ごとの転送レートを高めたい場合やレイテンシーをミリ秒単位に抑えたい場合は、Amazon CloudFront または Amazon ElastiCache を Amazon S3 のキャッシュとして使用します。

また、長距離間のクライアントと S3 バケットのデータ転送を高速化する場合は、Amazon S3 Transfer Acceleration を使用します。Transfer Acceleration は、CloudFront の世界中に点在するエッジロケーションを使用して、長距離間のデータ転送を高速化します。

Amazon S3 ワークロードで AWS Key Management Service (SSE-KMS) によるサーバー側の暗号化を使用している場合、ユースケースでサポートされるリクエスト率については、AWS Key Management Service デベロッパーガイドの「AWS KMS の制限」を参照してください。

以下のトピックでは、Amazon S3 を使用するアプリケーションのパフォーマンスを最適化するためのベストプラクティスガイドラインと設計パターンについて説明します。

このガイドラインは、Amazon S3 のパフォーマンスの最適化に関するこれまでのガイドラインよりも優先されます。例えば、Amazon S3 の以前のパフォーマンスのガイドラインでは、頻繁なデータ取得のパフォーマンスを最適化するために、ハッシュ文字列を使用してプレフィックスの命名をランダム化することを推奨していました。現在は、パフォーマンスを向上させるためにプレフィックスの命名をランダム化する必要はなくなり、プレフィックスに日付順の名前を使用できるようになりました。Amazon S3 のパフォーマンス最適化に関する最新情報については、パフォーマンスガイドラインおよびパフォーマンス設計パターンを参照してください。

Amazon S3 のパフォーマンスのガイドライン

アプリケーションで Amazon S3 の最適なパフォーマンスを得るために、以下のガイドラインを推奨しています。

トピック

- パフォーマンスを測定する
- ストレージ接続を水平にスケールする
- バイト範囲のフェッチを使用する
- レイテンシーの影響を受けやすいアプリケーションのリクエストを再試行する
- 同じ AWS リージョンで Amazon S3 (ストレージ) と Amazon EC2 (コンピューティング) を組み合わせる
- Amazon S3 Transfer Acceleration を使用して距離によるレイテンシーを最小限に抑える
- 最新バージョンの AWS SDK を使用する

パフォーマンスを測定する

パフォーマンスを最適化する際は、ネットワークスループット、CPU、および DRAM (Dynamic Random Access Memory) の要件を確認してください。これらのさまざまなリソースの要件の組み合わせに応じて、さまざまな Amazon EC2 インスタンスタイプを評価することをお勧めします。詳細については、Amazon EC2 Linux インスタンス用ユーザーガイドの「インスタンスタイプ」を参照してください。

これは、パフォーマンスの測定時に HTTP 分析ツールを使用して DNS ルックアップ時間、レイテンシー、およびデータ転送速度を確認する際にも役立ちます。

ストレージ接続を水平にスケールする

多くの接続間にリクエストを分散することが、パフォーマンスを水平にスケールする一般的な設計パターンです。パフォーマンスの高いアプリケーションを作成するには、Amazon S3 を従来のストレージサーバーのような 1 つのネットワークエンドポイントではなく、非常に大きな分散システムと考えます。最適なパフォーマンスは、複数の同時リクエストを Amazon S3 に発行することで実現できます。これらのリクエストを別々の接続に分散して、Amazon S3 の利用可能な帯域幅を最大化します。Amazon S3 には、バケットへの接続数に制限はありません。

バイト範囲のフェッチを使用する

GET Object リクエストで HTTP ヘッダー Range を使用すると、オブジェクトのバイト範囲を取得して、指定した部分のみを転送できます。Amazon S3 への同時接続を使用して、同じオブジェクトのさまざまなバイト範囲をフェッチできます。これにより、単一のオブジェクト全体のリクエストに対して高い集約スループットを実現できます。大きなオブジェクトの小さい範囲をフェッチすると、リクエストの中断時にアプリケーションで再試行回数の向上が可能になります。詳細については、「オブジェクトの取得」を参照してください。

バイト範囲リクエストの標準的なサイズは 8 MB または 16 MB です。マルチパートアップロードを使用してオブジェクトを PUT する場合、最適なパフォーマンスのために同じパートサイズで (少なくともパート境界に沿って整列されている) それらのオブジェクトを GET することをお勧めします。GET リクエストは、個々のパートを直接アドレス指定できます (例えば、`GET ? partNumber=N`)。

レイテンシーの影響を受けやすいアプリケーションのリクエストを再試行する

積極的なタイムアウトと再試行は、一定のレイテンシーの促進に役立ちます。Amazon S3 を大規模に利用している場合、最初のリクエストが遅くても、再試行のリクエストでは別のパスを選択してすぐに成功することがあります。AWS SDK には、特定のアプリケーションの許容値に調整できる設定可能なタイムアウト値と再試行値があります。

同じ AWS リージョンで Amazon S3 (ストレージ) と Amazon EC2 (コンピューティング) を組み合わせる

S3 のバケット名はグローバルに一意ですが、各バケットはバケットの作成時に選択したリージョンに保存されます。パフォーマンスを最適化するには、可能であれば、同じ AWS リージョンの Amazon EC2 インスタンスからバケットにアクセスすることをお勧めします。これにより、ネットワークレイテンシーとデータ転送費用を低減できます。

データ転送費用の詳細については、「Amazon S3 の料金」を参照してください。

Amazon S3 Transfer Acceleration を使用して距離によるレイテンシーを最小限に抑える

Amazon S3 Transfer Acceleration は、クライアントと S3 のバケットの長距離間の高速、簡単、安全なファイル転送を管理します。Transfer Acceleration は、Amazon CloudFront の世界中に点在するエッジロケーションを利用します。エッジロケーションに到着したデータは、最適化されたネットワークパスで Amazon S3 にルーティングされます。Transfer Acceleration は、大陸間で定期的にギガバイトからテラバイト単位のデータを転送するのに最適です。また、中央のバケットに対して世界中のお客様からアップロードが行われるクライアントにも役立ちます。

Amazon S3 Transfer Acceleration の速度比較ツールを使用すると、高速化した場合と高速化していない場合の Amazon S3 リージョン間でのアップロード速度を比較できます。速度比較ツールでは、マルチパートアップロードを使用して、ブラウザからさまざまな Amazon S3 のリージョンへのファイル転送を行い、Amazon S3 Transfer Acceleration を使用した場合と使用していない場合の比較が行われます。

最新バージョンの AWS SDK を使用する

AWS SDK では、Amazon S3 のパフォーマンスの最適化のために推奨されている多くのガイドラインが組み込みでサポートされています。SDK では、アプリケーションから Amazon S3 を利用するためのシンプルな API が提供されており、最新のベストプラクティスに従うように定期的に更新されます。たとえば、SDK は、HTTP 503 エラーでリクエストを自動的に再試行するロジックが含まれており、遅い接続に応答して適用するためにコードに投資しています。

また、SDK では Transfer Manager も提供されています。これは、接続の水平スケーリングを自動化し、必要に応じてバイト範囲のリクエストを使用して 1 秒あたりに数千ものリクエストを実現します。最新バージョンの AWS SDK を使用して最新のパフォーマンス最適化機能を取得することが重要です。

また、HTTP REST API リクエストを使用している場合は、パフォーマンスを最適化することもできます。REST API を使用する際、SDK の一部である同じベストプラクティスに従う必要があります。オブジェクトデータを同時フェッチできるように、リクエストが遅い場合のタイムアウトと再試行、および複数の接続を可能にしてください。REST API の使用については、Amazon Simple Storage Service API Reference を参照してください。

Amazon S3 のパフォーマンス設計パターン

Amazon S3 に対してストレージのアップロードや取得を行うアプリケーションを設計する場合は、アプリケーションの最適なパフォーマンスを実現するためにベストプラクティスの設計パターンを使用してください。AWS では、アプリケーションのアーキテクチャの計画時に考慮すべきパフォーマンスガイドラインも提供しています。

パフォーマンスを最適化するには、以下の設計パターンを使用します。

トピック

- 頻繁にアクセスされるコンテンツにキャッシュを使用する
- レイテンシーの影響を受けやすいアプリケーションのタイムアウトと再試行
- 高スループットのための水平スケーリングとリクエスト並列化
- Amazon S3 Transfer Acceleration を使用して長距離間のデータ転送を高速化する

頻繁にアクセスされるコンテンツにキャッシュを使用する

Amazon S3 にデータを保存するアプリケーションの多くは、ユーザーから繰り返しリクエストされる「作業セット」と言えるデータを提供します。ワークロードで一連のよく使用されるオブジェクトに対して繰り返し GET リクエストを送信する場合は、Amazon CloudFront、Amazon ElastiCache、AWS Elemental MediaStore などのキャッシュを使用してパフォーマンスを最適化することができます。キャッシュ導入が成功すると、レイテンシーが低くなり、データ転送速度が速くなります。また、アプリケーションでキャッシュを使用すると Amazon S3 にリクエストを直接送信する回数も減るため、リクエストにかかる費用を削減できます。

Amazon CloudFront は、各地に点在する一連の大規模な POP (Point Of Presence) で Amazon S3 のデータを透過的にキャッシュする高速なコンテンツ配信ネットワーク (CDN) です。複数のリージョンまたはインターネットからオブジェクトにアクセスする場合に CloudFront を使用すると、オブジェクトにアクセスするユーザーの近くにデータをキャッシュできます。これにより、Amazon S3 のアクセス数の多いコンテンツの配信パフォーマンスを高めることができます。CloudFront の詳細については、Amazon CloudFront 開発者ガイドを参照してください。

Amazon ElastiCache は、マネージド型のインメモリキャッシュです。ElastiCache を使用すると、オブジェクトをメモリにキャッシュする Amazon EC2 インスタンスをプロビジョニングできます。このキャッシュにより、GET レイテンシーが数桁減少し、ダウンロードスループットが大幅に向上します。ElastiCache を使用するには、アプリケーションのロジックを変更して、アクセス数の多い

オブジェクトをキャッシュに保存し、Amazon S3 にそのオブジェクトをリクエストする前にキャッシュを確認するようにします。ElastiCache を使用して Amazon S3 の GET のパフォーマンスを向上させる例については、ブログ記事の「Turbocharge Amazon S3 with Amazon ElastiCache for Redis」を参照してください。

AWS Elemental MediaStore は、Amazon S3 の動画ワークフローとメディア配信のために特別に作成されたキャッシュおよびコンテンツ配信システムです。MediaStore には、動画専用のエンドツーエンドのストレージ API が用意されており、パフォーマンスが重視される動画ワークロードに最適です。MediaStore の詳細については、AWS Elemental MediaStore ユーザーガイドを参照してください。

レイテンシーの影響を受けやすいアプリケーションのタイムアウトと再試行

アプリケーションが Amazon S3 から再試行が必要なことを示すレスポンスを受け取る場合があります。Amazon S3 は、バケット名とオブジェクト名を関連するオブジェクトデータにマッピングします。アプリケーションで発生するリクエスト率が高い場合 (通常、少数のオブジェクトに対して 1 秒あたり 5,000 リクエストを超える率が持続される)、アプリケーションは HTTP 503 slowdown レスポンスを受信することがあります。これらのエラーが発生した場合、各 AWS SDK はエクスポネンシャルバックオフを使用して自動再試行ロジックを実装します。AWS SDK を使用していない場合は、HTTP 503 エラーの受信時に再試行ロジックを実装する必要があります。詳細については、AWS 全般のリファレンスの「AWS でのエラー再試行とエクスポネンシャルバックオフ」を参照してください。

Amazon S3 は、処理を継続するための新しいリクエストレートに応じて自動的にスケールし、パフォーマンスを動的に最適化します。Amazon S3 が新しいリクエスト率のために内部的に最適化している間、最適化が完了するまで一時的に HTTP 503 リクエストレスポンスが送信されます。Amazon S3 が新しいリクエストレートに応じてパフォーマンスを内部的に最適化すると、リクエストはすべて再試行なしで通常どおり処理されます。

レイテンシーが重要なアプリケーションの場合、Amazon S3 では遅いオペレーションを追跡して積極的に再試行することをお勧めします。リクエストを再試行する際は、Amazon S3 に新しく接続して改めて DNS ルックアップを実行することをお勧めします。

可変サイズの大きいリクエスト (たとえば、128 MB 超) を実行する際、達成されるスループットを追跡し、リクエストのうち、遅い方から 5 パーセントを再試行することをお勧めします。小さいリクエスト (たとえば、512 KB 未満) を実行する際、レイテンシーの中央値が数十ミリ秒の範囲内であることが多い場合、2 秒後に GET または PUT オペレーションを再試行することをお勧めします。追加

の再試行が必要な場合のベストプラクティスはバックオフすることです。たとえば、2 秒後に 1 回目の再試行を発行し、さらに 4 秒後に 2 回目の再試行を発行することをお勧めします。

アプリケーションが Amazon S3 に固定サイズのリクエストを実行する場合は、それぞれのリクエストの応答時間はより一定になると考えられます。この場合、シンプルな戦略はリクエストのうち、遅い方から 1 パーセントを特定してそれらを再試行することです。1 回の再試行でもレイテンシーの低減において効果的でありことが多いです。

サーバー側の暗号化で AWS Key Management Service (AWS KMS) を使用している場合、ユースケースでサポートされるリクエスト率については、AWS Key Management Service デベロッパーガイドの「クォータ」を参照してください。

高スループットのための水平スケーリングとリクエスト並列化

Amazon S3 は大規模な分散システムです。その規模を活用できるように、並列リクエストを Amazon S3 のサービスエンドポイントに水平にスケールすることをお勧めします。このようなスケーリングのアプローチは、Amazon S3 でのリクエストの分散だけでなく、ネットワークで複数のパスに負荷を分散するためにも役立ちます。

転送のスループットを高めるために、Amazon S3 では複数の接続によってデータの GET や PUT を並列で行うアプリケーションを使用することをお勧めします。このような並列化は、AWS Java SDK の Amazon S3 Transfer Manager でサポートされています。また、その他のほとんどの AWS SDK でも同様の機能が提供されています。一部のアプリケーションでは、さまざまなアプリケーションスレッドで、またはさまざまなアプリケーションインスタンスで複数のリクエストを同時に起動することで、並列接続を実現できます。採用する最良のアプローチは、アプリケーション、およびアクセスするオブジェクトの構造によって異なります。

AWS SDK を使用して、AWS SDK で転送の管理を使用するのではなく GET リクエストまたは PUT リクエストを直接発行できます。このアプローチにより、ワークロードをより直接的に調整できると同時に、発生する可能性がある HTTP 503 レスポンスの再試行とその処理に引き続き SDK のサポートを活用できます。一般的なルールとして、リージョン内の大きなオブジェクトを Amazon S3 から Amazon EC2 にダウンロードする場合は、8 ~ 16 MB の粒度でオブジェクトのバイト範囲の同時リクエストを実行することをお勧めします。同時リクエストは、必要なネットワークスループットの 85 ~ 90 MB/秒ごとに 1 つ実行します。10 Gb/s のネットワークインターフェイスカード (NIC) を使用するには、個別の接続で約 15 の同時リクエストを使用します。より多くの接続で同時リクエストをスケールアップして、25 Gb/s や 100 Gb/s の NIC などのより高速な NIC を使用できます。

パフォーマンスの測定は、同時に発行するリクエスト数を調整する場合に重要です。一度に 1 つのリクエストから始めることをお勧めします。達成されるネットワーク帯域幅とデータの処理でアプリ

ケーションで使用されるその他のリソースの使用を測定します。その後、ボトルネックとなっているリソース (つまり、使用量が最も高いリソース) と有用である可能性が高いリクエスト数を特定できます。たとえば、一度に 1 つのリクエストの処理で CPU 使用率が 25 パーセントの場合、これは最大 4 つの同時リクエストに対応できることを示しています。

測定は不可欠であり、リクエスト率としてのリソース利用が向上していることを確認する価値があります。

アプリケーションで REST API を使用して Amazon S3 にリクエストを直接発行する場合は、HTTP 接続のプールを使用して、一連のリクエストで各接続を再利用することをお勧めします。リクエストごとの接続セットアップを回避すると、各リクエストで TCP スロースタートと Secure Sockets Layer (SSL) ハンドシェイクを実行する必要がなくなります。REST API の使用方法については、「Amazon S3 REST API の概要」を参照してください。

最後に、DNS に注意するとともに、リクエストが Amazon S3 の広範な IP アドレスのプールに分散されていることを確認することもお勧めします。Amazon S3 の DNS の問い合わせは、多数の IP エンドポイントのリストを確認します。ただし、キャッシュリゾルバ、または 1 つの IP アドレスを再利用するアプリケーションコードでは、アドレス多様性とそれによる負荷分散のメリットが得られません。コマンドラインツールの `netstat` などのネットワークユーティリティツールを使用すると、Amazon S3 との通信に使用されている IP アドレスを確認できます。また、使用する DNS 設定のガイドラインも提供しています。このガイドラインの詳細については、「リクエストルーティング」を参照してください。

Amazon S3 Transfer Acceleration を使用して長距離間のデータ転送を高速化する

Amazon S3 Transfer Acceleration は、グローバルに分散したクライアントと Amazon S3 を使用する特定のリージョンのアプリケーションの距離によるレイテンシーを最小限に抑える、またはなくすために有効です。Transfer Acceleration は、データの転送に CloudFront の世界中に点在するエッジロケーションを利用します。AWS エッジネットワークでは、50 を超えるロケーションに接続ポイントがあります。現在、このネットワークは、CloudFront でのコンテンツの配信や Amazon Route 53 に対する DNS の問い合わせへの迅速な応答のために使用されています。

このエッジネットワークは、Amazon S3 との間のデータ転送の高速化にも役立ちます。これは、大陸全域または大陸間でデータを転送する、高速なインターネット接続がある、大きなオブジェクトを使用する、またはアップロードするコンテンツが多数あるアプリケーションに最適です。エッジロケーションに到着したデータは、最適化されたネットワークパスで Amazon S3 にルーティングされ

ます。一般的に、Amazon S3 のリージョンから遠いほど、Transfer Acceleration による速度の向上が期待できます。

新しいバケットまたは既存のバケットで Transfer Acceleration をセットアップできます。離れた Amazon S3 Transfer Acceleration エンドポイントを使用して、AWS のエッジロケーションを使用することができます。Transfer Acceleration がクライアントのリクエストのパフォーマンスに役立つかどうかをテストする最も良い方法は、Amazon S3 Transfer Acceleration の速度比較ツールを使用することです。ネットワークの設定および条件は、随時、場所によって異なります。そのため、Amazon S3 Transfer Acceleration がアップロードのパフォーマンスを向上させることができると考えられる転送に対してのみ料金が発生します。さまざまな AWS SDK での Transfer Acceleration の使用については、「Amazon S3 Transfer Acceleration の例」を参照してください。

寄稿者

本書の作成における寄稿者

- Amazon S3、VP、Mai-Lan Tomsen Bukovec
- Amazon S3、シニアプリンシパルエンジニア、Andy Warfield
- Amazon S3、プリンシパルエンジニア、Tim Harris

ドキュメントの改訂

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。

update-history-change	update-history-description	update-history-date
Updated	技術的な正確性について確認	2021 年 3 月 10 日
初版発行	初版発行	2019 年 6 月 1 日

注意

お客様は、この文書に記載されている情報を独自に評価する責任を負うものとします。本書は、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤーまたはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または暗示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は、AWS 契約により規定されます。本書は、AWS とお客様の間で締結されるいかなる契約の一部でもなく、その内容を修正するものでもありません。

© 2020 Amazon Web Services, Inc. or its affiliates.All rights reserved.