Elastic Load Balancing（ELB）を評価するためのベストプラクティス

本ドキュメントはアマゾン ウェブ サービス（以下、AWS）の Elastic Load Balancing（エラスティック ロード バランシング、以下 ELB）サービスの機能やアーキテクチャについて説明しています。このベストプラクティスをご覧頂くことで ELB をテストしたり評価したりする場合によく陥る問題を回避できるようになります。このホワイトペーパーは、ELB を少しでも利用した経験をお持ちか、過去にソフトウェアまたはハードウェアのロードバランサを使った経験のある方を特に対象としています。

ELB の概要

ELB は複数の Amazon Elastic Compute Cloud（以下、 Amazon EC2）インスタンスにアプリケーションへのアクセスを自動的に分散します。ELB をセットアップすることで、１つのアベイラビリティゾーンまたは複数のアベイラビリティゾーンにある Amazon EC2 インスタンスに対してアクセスの負荷分散をすることができます。また、ELB を使うことで耐障害性に優れたアプリケーションを構築することが出来ます。さらにアプリケーションのトラフィック増加に対応可能なキャパシティをシームレスに提供します。

複数のアベイラビリティゾーンに Amazon EC2 インスタンスを配置することで耐障害性のあるアプリケーションを構築することが可能です。人的操作を減らして、より優れた耐障害性を実現するには、ELB を使うのがよいでしょう。ロードバランサの後ろに EC2 インスタンスを配置すると、複数のインスタンスや複数のアベイラビリティゾーンにわたって、ロードバランサが自動的にトラフィックを分散するため、耐障害性を高めることが可能です。

さらに ELB は Amazon EC2 インスタンスの状態を確認します。問題のある Amazon EC2 インスタンスを検出すると、ELB はそれらにトラフィックを割り当てません。その代りに、健全な EC2 インスタンスに負荷を割り当てます。もし複数のアベイラビリティゾーンに EC2 インスタンスをセットアップして、その内の１つのアベイラビリティゾーンの EC2 インスタンス全てに問題が起こった場合、ELB は別のゾーンにある健全な EC2 インスタンスにトラフィックを送ります。問題のある EC2 インスタンスが健全な状態に回復すると、ELB はそれらのインスタンスに再び負荷を分散します。また、ELB はそれ自体が耐障害性の高い分散型のシステムであり、高頻度に監視されています。

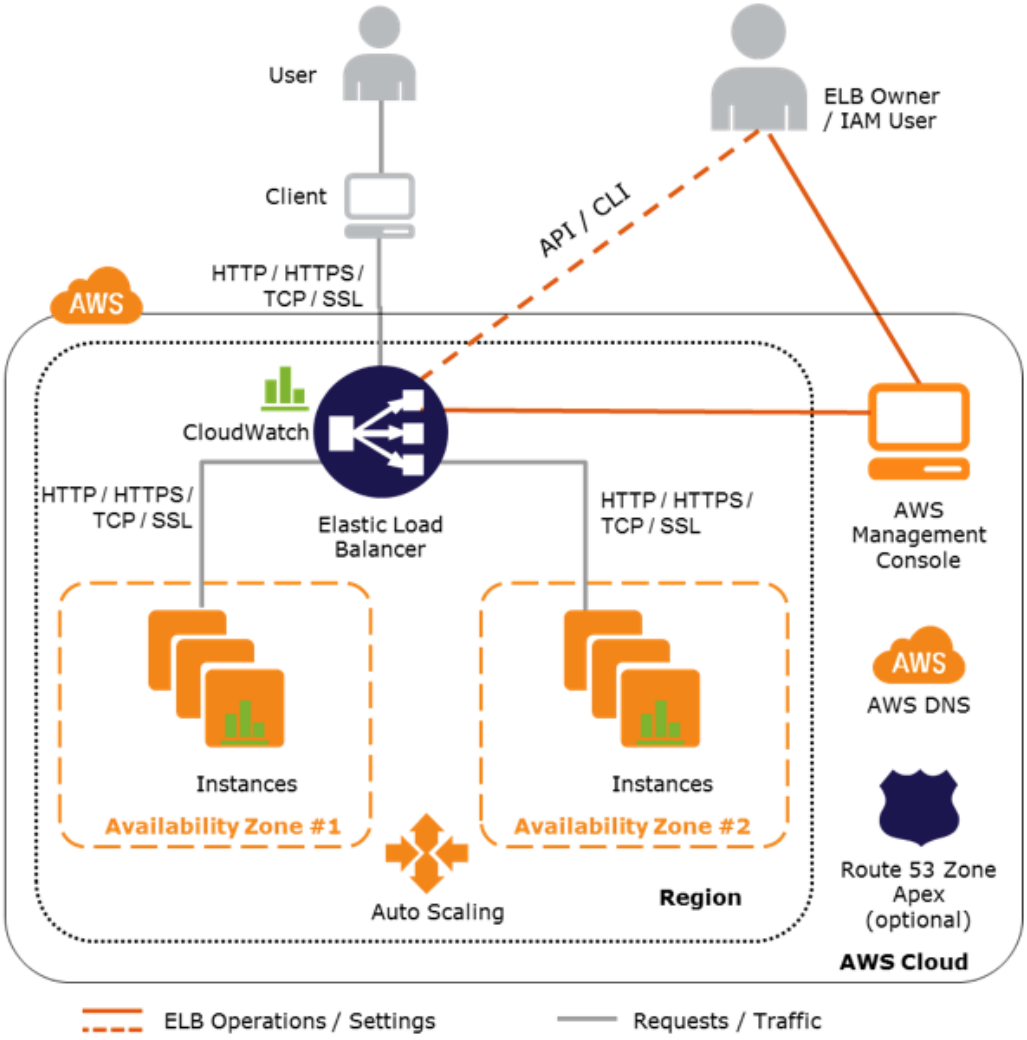
また、ELB は、変化するトラフィックに応じてバックエンドに十分なキャパシティを保つ機能を持ったオートスケーリングと統合することが可能です。例えばロードバランサの後ろにある EC2 インスタンスの数が 2 つ未満にならないようにしたい場合、オートスケーリ

ングにその条件をセットします。そうすれば、Amazon EC2 インスタンスが 2 つ未満になったことを検出すると、自動的に必要な数の EC2 インスタンスをオートスケーリンググループに追加します。もう一つの例をあげましょう。もしインスタンスのどれか一つのレイテンシーが 15 分間にわたって 4 秒を越えたら EC2 インスタンスを追加したい、といった場合に、そのような条件をセットすることが可能です。オートスケーリングはロードバランサの後ろで動作する時でさえも、EC2 インスタンスに対して適切なアクションをとります。もちろん、オートスケーリング自体は ELB を使っている、いないに関わらず、スケールを変化させることが可能です。

ELB の主なメリットの一つは、ロードバランサの管理、維持、スケーリングの複雑さを取り除いてくれることです。そのサービスは必要に応じて人手を介さずに自動的にキャパシティを追加したり削除したりできるよう設計されています。

ELB のアーキテクチャとその動作

ELB サービスのアーキテクチャには２つの論理的なコンポーネントがあります。ロードバランサとコントローラサービスです。ロードバランサはトラフィックを監視し、インターネットを経由してくるリクエストを処理します。コントローラサービスはロードバランサを監視し、必要に応じてキャパシティを増減させ、正しく動いているか確認します。



ロードバランサのスケーリング

ロードバランサを作成すると、流入してくるトラフィックを受け入れて Amazon EC2 インスタンスへリクエストを送るように設定する必要があります。これらの設定パラメータはコントローラによって保存され、全ロードバランサが設定に基づいて正しく動作するように制御します。コントローラは、ロードバランサも監視し、クライアントのリクエストを処理できるようキャパシティを管理します。キャパシティを増やす際には、より大きなリソース（より高いパフォーマンス特性を持つリソース）を用いるか、より数多くの個別リソースを利用します。ELB サービスでは、ロードバランサがスケールする時にはロードバランサのドメインネームサービス（DNS）のレコードを更新します。すると新しいリソー

スは DNS に登録された個別の IP アドレスを持つようになります。作成された DNS レコードは 60 秒の Time-to-Live(TTL)の設定を持ち、クライアントが少なくとも 60 秒ごとに DNS を再度検索するよう想定されています。デフォルトでは ELB はクライアントが DNS 名前解決を実行するときには、それぞれの DNS 名前解決リクエストごとにランダムな順番で複数の IP アドレスを返します。トラフィックの状況が変化すると、コントローラサービスは、より多くのリクエストを処理できるように、全てのアベイラビリティゾーンで均等にスケールさせます。

ロードバランサは ELB で登録されている Amazon EC2 インスタンスのヘルスチェックも実行します。インスタンスがサービス中であり健全であると見なされるまでに、ELB の設定の中で定義された対象に対してヘルスチェックが行われ、定義された回数の分だけヘルスチェックが成功する必要があります。例えば、ELB で登録された複数のインスタンスに対して、仮にヘルスチェックの間隔を 20 秒に設定し、ヘルスチェックの成功回数を 10 回に設定した場合、ELB がトラフィックをインスタンスに送るようになるまでに少なくとも 200 秒かかります。また、ヘルスチェックでは失敗回数のしきい値を設定できます。例えば、ヘルスチェックの間隔を 20 秒に設定し、失敗回数のしきい値を 4 回にした場合、インスタンスがリクエストに反応しなくなり、そのインスタンスがサービス停止中と見なされるまでに 80 秒かかります。しかし、インスタンスが終了したときには登録解除はすぐに実行されるので問題ありませんが、インスタンス登録解除までに遅延が起こる場合もあり得ます。このため、それらのインスタンスを終了させる前にインスタンスを登録解除することが重要です。もし登録が解除される場合、インスタンスは非常に短い時間でサービスから削除されます。

負荷テストシナリオの計画

多くの開発者は仮想プライベートサーバ（VPS）または物理ハードウェア環境の中で負荷テストツールを使ったことがあり、このような状況でロードバランサの動作をどのように評価するのかを理解しています。彼らはロードバランサを評価して、次のような決断をします。

1. さまざまなトラフィックレベルをサポートするために何台のアプリケーションサーバを必要とするのか？
2. レスポンスタイムを低下させることなくトラフィックを分散させるには何台のロードバランサが必要なるのか？
3. ハードウェアやネットワーク障害のイベント時にアプリケーションが動作し続けることが可能か？

これらのすべての質問は重要ではありますが、アプリケーションの構成の仕方によっては、上記質問の１、もしくは 2 の質問は Amazon EC2 環境では重要ではないかもしれません。

もしオートスケーリングを使っていて、アプリケーションサーバを必要に応じて追加したり削除したりするならば、前もって上記の決断をしなくても良くなります。ELB を使っている時に２の質問を考慮する必要はありません。なぜなら ELB が動作している時間とそれが処理するデータに対してのみ支払い、アプリケーションを提供するために必要な実際のキャパシティに対しては支払う必要がないためです。

ELB の目的はアプリケーションの要求に合致するようスケールすることであり、これはロードバランサ用の DNS レコードを更新することによって実現されているため、ELB サービスのいくつかの特長がテストシナリオにどのように影響するかを意識する必要があります。

DNS 解決

ロードテスト用のツールにはたくさんの種類があり、それらのツールのほとんどは、サーバが処理することができるトラフィックの量に基づき何台のサーバが必要とされるかという要求対処できるよう設計されています。この状況でサーバの負荷をテストするには、サーバがいつ飽和するかを決めるためにトラフィックを増やし、飽和点に影響を与える要素を決められるようなリクエストとレスポンスのサイズに基づいたテストを繰り返し実行します。

ロードバランサを作成するときには、キャパシティのデフォルトのレベルが割り当てられ構成されます。ELB のトラフィックにおける変化を見ると、それはスケールアップしたりダウンしたりします。ELB がスケールするために必要な時間は 1 分から 7 分程度であり、トラフィックのプロファイルにおける変化に依存します。ELB がスケールするときには、IP アドレスの新しいリストで DNS レコードを更新します。クライアントが増加したキャパシティを活用するために、ELB は DNS レコードで TTL を 60 秒に設定しています。テストの中に DNS レコードの変更を組み込むことは重要です。増加した負荷をシミュレートするために、DNS の再解決と複数のテストクライアントの利用を確認しましょう。そうしないと、実際に ELB がより多くの IP アドレスを割り当てたときに、そのテストはずっと一つの IP アドレスに向かって負荷をかけ続けるかもしれません。実際のエンドユーザの場合は一つの IP アドレスを常に利用するわけでは無く、割り当てられた複数の IP アドレスのいずれかを利用するので、このテストの動作はあまり参考にはならないでしょう。

スティッキーセッション

ELB は cookies を使用した形でのスティッキーセッション（セッションアフィニティとしても知られています）をサポートする機能を持っています。デフォルトではこれらの機能は無効になっており、簡単な変更で有効化できます。使用できるスティッキーセッションのポリシーは 2 種類ありますが、その効果は同じではありません。スティッキーセッションが有効化された ELB を使用すると、ユーザがアプリケーションに継続してアクセスする

ようにバックエンドの同じインスタンスにトラフィックが送られます。負荷テストにスティッキーセッションを使うように設計するとき、この特徴をどのようにテストで使用するかを決めることが重要です。スティッキーセッションが負荷テストおよび実践的な場面世界の両方において、どのように課題となりえるかを考えてみてください。

例えば、通常サイトを提供するのに４つのインスタンスを使っているアプリケーションを考えてみます。それぞれのユーザはそのインスタンスのうちの１つに 10 分継続してリクエストを送るような指示を ELB 用に含めた Cookie を受け取ります。トラフィックにおける大きな変化が生じ、サイトにやってくるユーザの数が 3 倍になったとき、これらの新しいユーザの全ては４つのインスタンスのうちの１つにリクエストを向ける Cookie を取得します。バックエンドにどれだけキャパシティを追加しても、たくさんのユーザはその cookie が期限切れになるまで元々の４つのインスタンスにリクエストを送ります。

スティッキーセッションは強力な機能ではあるかもしれませんが、対象となるアーキテクチャにどのように合わせるか、そしてアプリケーションのシナリオにより適したソリューションであるかどうかを考えることが重要です。

ヘルスチェック

ELB はあなたが設定した構成を使って、バックエンドのインスタンスにヘルスチェックを実行します。ELB でインスタンスを登録したときに、ヘルスチェックが特定回数無事完了するまでは、インスタンスは健全であると見なされません。また不成功に終わったヘルスチェックが特定回数に到達すると、そのインスタンスへはトラフィックを送らないようになります。（ですが、ELB には登録されてはいます。）ELB に登録されている限りは、ロードバランサはインスタンスにヘルスチェックし続けます。ヘルスチェックの間隔を長く設定する、もしくは、健全であると見なされるためのヘルスチェックの成功回数のしきい値を多く設定すると、インスタンスが ELB からトラフィックを受け取り始めるまでにより時間がかかります。これは、バックエンドのアプリケーションインスタンスに必要なキャパシティを追加する（オートスケーリングなどの）自動処理やシステムを使う場合は特に重要です。

アーキテクチャと負荷テストを計画するのと同様に、どのようにヘルスチェックを管理するかを決めることは重要です。Web サーバが反応しているかを頻繁にチェックする軽量のページをヘルスチェックで利用するか、または頻繁にヘルスチェックはしないがアプリケーションすべてのパーツが適切に機能しているかをチェックする重たいページをヘルスチェックで利用するかどうかを考えてみましょう。ELB からインスタンスに向けて行われるすべてのヘルスチェックリクエストは Amazon CloudWatch の監視結果には表示されませんが、アプリケーションのログには（ログが有効化されている場合）は書き込まれることになります。あなたが構成したヘルスチェックに加えて、ELB はインスタンスに対して接

続することのみのチェックを頻繁に実施しています。このチェックはインスタンスの健康状態を決定するために使用されるわけではありませんが、登録したインスタンスが終了したときにサービスが検知することを保証するために使われています。

バックエンドの認証

SSL 暗号化がバックエンドのインスタンスとのコミュニケーションするときに使われる場合は、バックエンドの認証を任意で有効にすることが出来ます。この認証が構成されたときには、証明書の公開鍵が構成済みの公開鍵のホワイトリストに含まれている時のみバックエンドのインスタンスとコミュニケーションします。この強力な機能はさらなるレベルの保護とセキュリティを提供しますが、リクエストスループットはわずかに低くなることも意味します。そのレスポンス時間によっては、この特徴はあなたのアプリケーションにとって最適である場合もあれば、そうではない場合もあるかもしれません。

テストフレームワークの選択

テストのセットアップの詳細にかかわらず、負荷テストを実行するためにいくつかのソフトウェアやフレームワークを必要とするでしょう。負荷テストにおいては、時間をかけてリクエストの数を増やすことを考慮する必要があります。そして複数のクライアントまたはエージェント間でリクエストを分散する必要があるかもしれません。多くの負荷テストツールは別々の負荷生成クライアントをサポートしているものもあれば、テストの実装の調整を必要とする負荷テストツールもあるでしょう。テストに使われる３つのシナリオは次のセクションで記述しています。

シングルクライアントテスト

シングルクライアントテストでは同じクライアントを使ってサーバへのリクエストを生成します。ほとんどの場合においては、テストが初期化された後は名前の再解決はされません。この種類のテストの良い例は Apache Bench(ab)です。もしこの種類のテストツールを使っているならば、２つのオプションがあります。

1. ある特定のサンプルサイズとタイミングを持つ、個別に稼働することができるテストを書きます。そしてこれらのテストを分けて起動します。そうすればクライアントは名前の再解決（オペレーティングシステムや他の要素に依存する名前の再解決を強要することも可能でしょう）を行うでしょう。
2. 複数のクライアントインスタンスを起動しておよそ同じときに異なるインスタンス上でテストを初期化してください。AWS CloudFormation を使って、負荷テストスクリプトを複数のクライアントで起動し、SSH でリモートからコマンドを実行して、テストを初期化することができます。より多くの負荷生成クライアントをオ

ートスケーリングによって立ち上げることも出来ます。（例えば他の負荷生成クライアントの高負荷になって、平均 CPU が特定の閾値に達したときに、負荷生成クライアントの数を増やすなど）

しかし、生成しようとしている負荷によりますが、あなたのテストはクライアント側のリクエスト生成能力によって制約を受けるかもしれません。この場合、分散テストを考える必要があります。

組み込みのマルチクライアントテスト

いくつかのテストフレームワークはテストを実行するために複数のクライアントを生成します。一つの例は curl-loader というオープンソースソフトウェアのパフォーマンステストツールです。それぞれのクライアントは独立したプロセス内で動作するため、自分で DNS 再解決します。シングルクライアントテストのようにテストするための負荷がクライアントを使って生成できる負荷よりも大きいならば分散テストを考える必要があるかもしれません。

分散テスト

たくさんのクライアントを使って ELB をテストするために、Web サイトのテスト用の自動テストを実行することが可能なサービスプロバイダーがあります。アプリケーションによっては、この方法がアプリケーションの負荷テストをするのに最も簡単で効率的な方法かもしれません。この種類のサービスが合わない場合は、分散テストに役立つツールを考慮する必要があるかもしれません。例えばテストを実行しその結果をテストコントローラへ返してくれるようなテストクライアント（オープン疎ソースの Fabric フレームワークなど）を Amazon EC2 環境に起動する方法があります。分散テストが必要となることは多くありますが、特にとても高い負荷の場合は、分散テストフレームワークを使って実際のトラフィックをシミュレートすることは最も効果的なアプローチかもしれません。

推奨テストアプローチ

ELB テストのプランの中にはいくつかの特別な検討事項や取り組み方あります。その飽和点やブレーキングポイントのみならず、予測している”通常の”トラフィックのプロファイルもまたテストすることは重要です。もしサイトが通常のビジネス時間の間にトラフィックの一貫性のあるフローを受け取るような場合、トラフィックの低い場合と高い場合の落ち着いた状態をシミュレートすべきです。また低い時から高くなる時、またはその逆も同様にトラフィックのプロファイルの移行があるような朝や夜の時間帯をシミュレートすべきです。短い時間でトラフィックが重大な変化を受け取ると予測するならば、これをテストすべきです。アプリケーションが内部のイントラネットのサイトのものであれば、おそ

らくトラフィックは 1000 ユーザから 100,000 ユーザまで 5 分間で増加することは起こりえないでしょう。ですが、もし瞬間的なトラフィックを持つアプリケーションを構築しているのでしたら、この環境でテストして ELB がどのように動作するかだけでなく、アプリケーションとアプリケーション内部のコンポーネントがどのように動作するかを理解することが重要です。

テストの増強

いったんテストツールを配置したら、負荷の増やし方を定義する必要があるでしょう。 5 分おきに 50 パーセント未満のレートで負荷を増やすことが推奨されています。負荷生成のための段階的なパターンおよび線形的なパターンは ELB でうまく機能します。もしランダムの負荷生成を利用するつもりならば、急激な負荷の上限を設定して、ELB がスケールするまでの間、その上限を越えないようにすることが重要です。（詳細は ELB の暖気運転の項目をご覧ください）

アプリケーションのアーキテクチャがロードバランサでのセッションアフィニティを含むならば、以下に挙げるようないくつかの追加構成ステップが必要です。

スティッキーセッションを伴う負荷テスト

あなたの構成がスティッキーセッションを活用しているならば、複数の負荷生成クライアントを使うことが重要です。そうすれば ELB は実際の動きと同様に動作することが出来ます。もしこれらの調整をしなければ、ELB が負荷に対応するためにスケールする前に同じバックエンドのサーバに ELB はリクエストを送り続けて、そのサーバのみの負荷を高めることになるでしょう。このシナリオでテストするには、負荷を生成する複数のクライアントを使うテストソリューションを用いる必要があるでしょう。

環境の監視

ELB のメリットの一つは、Amazon CloudWatch によっていくつかのメトリクスが提供されていることです。負荷テストを実施している間で、監視すべき重要な 3 つのエリアがあります。ロードバランサ、負荷生成クライアント、そして ELB で登録されたアプリケーションインスタンスです。（アプリケーションが依存している EC2 インスタンスも同様です。）

ELB の監視

ELB は Amazon CloudWatch によって次のようなメトリクスを提供します。（ドキュメントでは http://aws.amazon.com/jp/documentation/cloudwatch/ でこれらのメトリクスの詳細な情報を提供しています。）

- 遅延時間

- リクエスト数
- 健全なホスト数
- 不健全なホスト数
- バックエンドからの 2xx-5xx のレスポンス数
- ELB からの 4xx-5xx のレスポンス数

アプリケーションをテストするとき、これらのメトリクスの全てを見ることは重要です。特に興味深い項目は ELB の 5xx レスポンス数、バックエンドの 5xx レスポンス数、そして遅延時間でしょう。

負荷生成クライアントの監視

また、負荷テスト中に負荷が正しく送信されて、そのレスポンスが正しく受け取ることができていることが保証できるように複数の負荷生成クライアントを監視することは重要です。もし Amazon EC2 の中でシングルテストクライアントを動作させているならば、負荷を処理するために十分スケールしていることを確認してください。より高い負荷においては、複数のクライアントを使うことは重要です。なぜなら、そのクライアントがネットワークの帯域上の制限を受けている場合に、テストの結果に影響を与えているかもしれないためです。

アプリケーションインスタンスの監視

他のテストで監視するのとまさに同じ方法であなたのアプリケーションを監視することを推奨します。代表的なツールをご利用頂けます。例えば Linux の top コマンドや Windows 用のログパフォーマンス分析（PAL）ですが、インスタンス用に Amazon CloudWatch を使うことも出来ます。（Amazon CloudWatch のドキュメントは http://aws.amazon.com/documentation/cloudwatch/をご覧ください。）テスト中にインスタンスの動作をよく見ることが可能であることを保証するために、詳細な 1 分ごとのメトリクスをアプリケーションインスタンスのために有効にするのも良いでしょう。

ELB テスト時の一般的な落とし穴

ELB のキャパシティのリミットへの到達

ELB は真のキャパシティリミットに到達することは稀ですが、メトリクスに基づいてスケールするまでの間に、ロードバランサがこれ以上リクエストを処理することが出来ない時、HTTP 503 エラーを返す期間があり得ます。ロードバランサはすべてのリクエストを待ち行列に入れようとはしません。そのため、ロードバランサがフル稼働である場合、追加のリクエストは失敗するでしょう。もしトラフィックが時間をかけて大きくなれば、この動作は正しく働きます。急激にトラフィックが増大する場合、または特殊な負荷テストシナリ

オの場合、ELB がその負荷に対応するためにスケールできるようになるよりも早い速度でトラフィックが送信されるかもしれません。この状況に対処するには以下に記述した２つの選択肢があります。

ロードバランサの暖気運転（Pre-Warming）

リクエストに応じて、ELB は予測されるトラフィックに基づいた最小スケールを持つように構成することが出来ます。瞬間的にトラフィックの増大が見込まれる場合、DDOS 攻撃がロードバランサ（またはお客様のインスタンス）に対して開始されるとき、または負荷テストが徐々にトラフィックが増加するよう負荷テストを構成できないときにこちらは必要となります。もしアプリケーションにとって典型的ではない極端な負荷での動作を評価するための負荷テストをするようでしたら、AWS サポートに同意してもらい、サポートにロードバランサの暖気運転をさせるよう依頼することを推奨します。テストの開始日、終了日、さらに秒間で予測されるリクエストの速度とテストする典型的なリクエスト／レスポンスの合計サイズを提供する必要があります。

負荷テストパラメータの管理

5 分間隔で 50%以上のトラフィック増加をしないように負荷テストを設定することが推奨されます。テストは段階的に、もしくは緩やかなカーブでトラフィックを増大させる場合のどちらでもセットアップ可能です。大量の負荷にまで増加させる場合でさえも、ELB は要求に応えるためにスケールするでしょう。

名前の再解決

クライアント側で少なくとも 1 分に 1 回名前の再解決がされない場合、ELB によって DNS に追加された新しいレコードがクライアント側で使われないことがあるでしょう。これは、ELB 全体では過負荷になってはいないものの、クライアントが ELB で配置されているリソースの一部のみに負荷をかけ続けることを意味します。これは実際のシナリオでは起こりうる問題ではありませんが、名前の再解決がされないテストツールのために、クライアント側で頻繁に名前の再解決をしなければいけない問題になる可能性があります。

スティッキーセッションと一意的なクライアント

スティッキーセッションが有効である場合、負荷テストフレームワークがテストの中でコネクションとクライアントを再利用しないことよりも、むしろそれぞれのリクエストで一意のクライアントを作成することの方が重要です。もしこれが実施されない場合、そのクライアントは利用可能なキャパシティのごく一部に負荷をかけしながら、同じ ELB リソースと同じバックエンドインスタンスにトラフィックを常に送るでしょう。

初期のキャパシティ

ELB には初期のキャパシティのスタートポイントがあり、トラフィックに応じてスケールアップまたはダウンします。いくつかのパターンにおいては、ロードバランサに最初に届くトラフィックの量が、ロードバランサの初期のキャパシティ構成で対応可能なトラフィックの量よりも大きいでしょう。あるいは、ロードバランサが作成されてしばらくの間使われていない場合（一般的には数時間ですが、可能性としては早ければ 1 時間で）トラフィックがロードバランサに到達する前にロードバランサがスケールダウンするかもしれません。これは初期のキャパシティレベルに届くように、ロードバランサをスケールアップさせなければいけないことを意味します。

コネクションタイムアウト

ELB で登録されるインスタンスを構成するときには、これらのインスタンスのアイドルタイムアウトを少なくとも 60 秒に設定することが重要です。ELB は 60 秒後にアイドルコネクションをクローズします。、バックエンドのインスタンスが少なくとも 60 秒のタイムアウトを持つことを想定しています。もしバックエンドのインスタンス側のタイムアウトを 60 秒以上に設定しなかった場合、ELB はそのインスタンスを間違って不健全なホストとして見なして、そのインスタンスにトラフィックを送ることを止めてしまうかもしれません。 j

参考文献

ELB ページ

- http://aws.amazon.com/elasticloadbalancing
- http://aws.amazon.com/documentation/elasticloadbalancing

Amazon CloudWatch ページ

- http://aws.amazon.com/cloudwatch
- http://aws.amazon.com/documentation/cloudwatch/

オートスケーリングページ

- http://aws.amazon.com/autoscaling
- http://aws.amazon.com/documentation/autoscaling

分散テストソリューション

分散負荷テストのための有望なソリューションは多くあります。以下はあなたのテストの必要性を満たすことができるかもしれないパートナーの一部のリストです。さらにソリューションをお求めの場合は AWS サイトのソリューションプロバイダーセクションをご覧ください。http://aws.amazon.com/solutions/solution-providers

NeoTys: http://aws.amazon.com/solutions/solution-providers/neotys/

SOASTA: http://aws.amazon.com/solutions/solution-providers/soasta/

Apica: http://aws.amazon.com/solutions/solution-providers/apicasystems/

Eviware: http://aws.amazon.com/solutions/solution-providers/eviware/