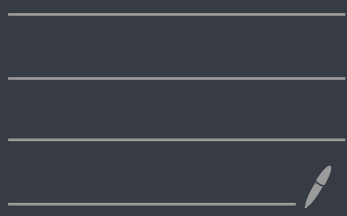

ELB

AWSでは多くのインスタンスタイプが用意されているので、ある程度まではインスタンスタイプを上げるとこで対応できる。このような垂直にスペックを上げる対応をスケールアップと呼ぶ。

しかし、スケールアップには限度があり、いずれインスタンスタイプの上限とぶつかってしまう。

また、単一インスタンスで運用を続けると、「そのインスタンスの停止=サービス全体の停止」という状況になる。

このような問題を単一障害点(Single Point Of Failure, SPOF)と呼び、単一障害点のある設計は推奨されない。

それに伴い、Webサーバーレイヤーでベストプラクティスとなっているのが、EC2インスタンスを水平に並べるスケールアウトである。

EC2インスタンスを複数並べ、その前段にロードバランサーを配置してリクエストを各インスタンスに分散させる。

ロードバランサーとしてはEC2上にBIG-IPといった製品を導入することもできるが、ロードバランサーのマネージドサービスであるElastic Load Balancing(以下ELB)を利用するのがおすすめ。

ELBの種類

ELBには下記の3タイプのロードバランサーがある。

- Classic Load Balancer(CLB): L4/L7レイヤーでの負荷分散を行う
- Application Load Balancer(ALB): L7レイヤーでの負荷分散を行う
- Network Load Balancer(NLB): L4レイヤーでの負荷分散を行う。HTTP(S)以外のプロトコル通信の負荷分散をしたいときに利用する。

CLBとALBは同じアプリケーションレイヤーでの負荷分散を行うが、後継サービスであるALBの方が安価で昨日も豊富である。具体的な機能としては、WebSocketやHTTP/2に対応していること、URLパターンによって振り分け先を変えるパスベースルーティング機能が提供されていることなどが挙げられる。

NLBは、HTTP(S)以外のTCPプロトコルの負荷分散を行うために用いられる。

ロードバランサー自体が固定のIPアドレスを持つなどの特徴がある。

ELBの特徴

マネージドサービスであるELBを用いるメリットとして、ELB自体のスケーリングが挙げられる。

EC2インスタンス上にロードバランサーを導入する場合は、そのインスタンスがボトルネックにならないように設計する必要がある。それに対してELBを用いた場合、負荷に応じて自動的にスケールする設計になっている。

注意点として覚えておくべきこととして、このスケーリングが瞬時に完了するわけではない点がある。

そのため、数分程度で急激に負荷がかかるシーン、では間に合わないことがある。

もしこのような急激な負担増(スパイク)が予想できる場合は、事前にELBプレウォーミングの申請をすること。

もう一つ、ELBの大きなメリットとしてヘルスチェック機能があること。

ヘルスチェックは、設定された間隔ではいかにあr配下にあるEC2にリクエストを送り、各インスタンスが正常に動作しているかを確認する機能である。もし以上なインスタンスが見つかった場合は自動的に切り離し、その後正常になったタイミングで改めてインスタンスをELBに紐づける。

ELBのヘルスチェックには下記の設定値がある。

- 対象のファイル(例: /index.php)
- ヘルスチェックの間隔(例: 30秒)
- 何回連続でリクエストが失敗したらインスタンスを切り離すか(例: 2回)
- 何回連続でリクエストが成功したらインスタンスを紐づけるか(例: 10回)

もし、Webサーバーだけでなく、DBサーバーまで正常に応答することを確認したい場合は、DBにリクエストを投げるファイルをヘルスチェックの対象ファイルにする。また、上記の設定だと紐付けまで30秒 * 10回 = 5分かかってしまうのでもう少し短い時間で紐付けを行いたい場合は、ヘルスチェックの間隔を短くするなどの調整を行う。

Auto Scaling

Auto Scalingは、システムの利用状況に応じて自動的にELBに紐づくインスタンスの台数を増減させる機能である。

インフラリソースを簡単に調達でき、そして不要になれば使い捨てできるクラウドならではの機能である。

Auto Scalingでは次のような項目を設定することで、自動的なスケールアウト/スケールインを実現する。

- 最小のインスタンス数(例: 4台)
- 最大のインスタンス数(例: 10台)
- インスタンスの数を増やす条件と増やす数(例: CPU使用率が80%を超えたら2台増やす)
- インスタンスの数を減らす条件と減らす数(例: CPU使用率が40%を割ったら2台減らす)

また、インスタンスの数を減らす際にどのインスタンスから削除するか、たとえば「起動時間が最も過去となる古いインスタンスから削除」といった設定もできる。

Auto Scalingを利用することで、繁忙期やピーク時期はインスタンス数を増やし、閑散期や夜間のリクエストが少ない時はそれなりのインスタンス数でサービスを運用することも可能。

スケーリングポリシー

Auto Scalingのスケーリング方法は、大きく3つに分類できる。

- 動的なスケーリング
- 予測スケーリング
- スケジュールスケーリング

頻繁に利用されるスケーリングには、さらに3つのスケーリングポリシーがある。

- 簡易スケーリング
- ステップスケーリング
- ターゲット追跡スケーリング

簡易スケーリング

1つ目の簡易スケーリングは、CPU使用率が70%を超えたらといったように1つのメトリクスに対して1つの閾値を設定する。

最初期からあるスケーリング方法で、現在ではひすいしょう非推奨となっている。

簡易スケーリングは、次に紹介するステップスケーリングで代用できるので、そちらを使うようにする。

ステップスケーリング

ステップスケーリングは、1つのメトリクスに対して複数の閾値を設定する。

例えば、CPU使用率が50%を超えた場合、60%を超えた場合と、段階(ステップ)ごとの設定ができる。

このように複数の閾値を設定できるが、1つの閾値のみを設定することもできる。

つまり、簡易スケーリングの上位互換ということである。そのため、AWSとしては簡易スケーリングよりステップスケーリングを推奨している。

ターゲット追跡スケーリング

ターゲット追跡スケーリングは、1つのメトリクスに対して目標値を設定する。

例えば、「CPUの利用率を50%に」という目標値を設定すると、Auto Scalingグループ全体でCPU利用率が50%を維持できるように自動的に調整される。ステップスケーリングのように細かく刻んで設定しなくてもAWS側がコントロールしてくれる。

今後は、このターゲット追跡スケーリングが主流になる見通し(2020年現在)

スケーリングの設定をする際には、起動設定や起動テンプレートの設定が必要である。

実際に手を動かしてみて、設定の仕方を習得するのが重要。

スケールアウトの猶予時間・ウォームアップ・クールダウン

システムのパフォーマンスや信頼性を高めるには、Auto Scalingを使って需要に応じてインスタンス数を増減させる必要がある。

その際には、インスタンスの立ち上がり中に新たなインスタンスが立ち上がることを抑止する必要がある。

そうしないと、想定以上の台数になってしまう可能性があるからである。AWSにはそのための機構がいくつかある。

スケールアウトの猶予期間・ウォームアップ・クールダウンである。

― 猶予期間 ―

まず最初に、ヘルスチェックと猶予期間である。

ヘルスチェックは、Auto Scalingの管理対象下にあるインスタンスの動作をチェックして、異常が認められた場合は新しいインスタンスと置き換えられる。

インスタンスの起動時間やインスタンス内のプロセスの起動時間があるために、すぐにそのインスタンスが利用可能な状態になるわけではない。その間にヘルスチェックがエラーが出続けると想定したスケーリング動作にならないので、一定期間ヘルスチェックをしないという猶予期間がある。

— ウォームアップとクールダウン —

猶予期間以外にも、ウォームアップとクールダウンというパラメータがある。

まずはクールダウンで正式名称は、スケーリングクールダウンという。

これは、Auto Scalingが発動した直後に、追加でインスタンスの増減がなされるのを防ぐための設定である。

猶予期間はヘルスチェックの猶予であり、クールダウンはインスタンスの増減のアクションの発動に対してのパラメータとなる。

ウォームアップは前述の通り、おおむねステップスケーリングにおいて差分でインスタンスを追加するための機能である。

CPUの利用率に応じて、設定したポリシーに沿ってインスタンスを追加する。

ウォームアップの動作のポイントは、2つである。1つ目はウォームアップ期間中には、そのアラートでのインスタンス追加はされない点。

2つ目は、ウォームアップ期間中に次のアラート(70%)が発動した場合、差分の台数が追加される点。

ウォームアップは、ステップスケーリングとターゲット追跡スケーリングポリシーに対応している。

その他のAuto Scalingのオプション

― ライフサイクルフック ―

ライフサイクルフックとは、Auto Scalingグループによるインスタンスの起動時、または削除時にインスタンスを一時停止してカスタムアクションを実行する機能である。

例えば、起動時にデータを取得してきたり、削除時にログやデータの退避の処理を追加するといった用途で使える。

待機時間はデフォルトで1時間、最大で48時間まで設定できる。

― 終了ポリシ ―

負荷に応じでインスタンスを増やすことをスケールアウトという。これに対して、負荷が下がってインスタンスが減る場合はスケールインという。

Auto Scalingで、スケールインの際にどのインスタンスを削除するのかを決めるのが終了ポリシーである。

デフォルトの設定では、インスタンスの数がアベイラリティゾーンに均等に配分されるように削除される。

ELBとAuto Scalingを利用する際の設計ポイント

ELBとAuto Scalingを導入する際に気をつけたいポイントが2つある。

1つ目は、サーバーをステートレスに、つまり、状態を保持し内容に設計をすることである。
データはデータベースに格納し、ファイルはインスタンス外のS3に置くといった設計に変更すると、スケールインのタイミングでインスタンスが削除されてもデータを参照することができる。

2つ目は、AZをまたがってインスタンスを配置する設計にすべきという点である。
AWSではごく稀にAZ全体の障害が発生することがある。その時にインスタンスが1つのAZに固まっていると、すべてのインスタンスが入り用できなくなり、結果としてサービス全体が停止してしまう。

このように、AWSでは「いかに単一障害点をなくすか」、「部分の障害は起こり得ることを前提として設計する」という思想が非常に大切になってくる。