シャッフルシャーディングを使ったワークロードの分離

Colm MacCárthaigh

シャッフルシャーディングを使ったワークロードの分離

現在では、世界でも最も大きいビジネスやほとんどの著名なウェブサイトをホスティングしている Amazon Route 53 ですが、その立ち上がり時期においては、はるかに控え目なものでした。

DNS ホスティングの対応開始

AWS のサービス開始後、さほど長い時間が経過する前に、AWS のお客様からは、Amazon Simple Storage Service (S3) 、Amazon CloudFront 、Elastic Load Balancing のサービスをドメインのルートで使用して、「www.amazon.com」だけでなく「amazon.com」というドメイン名も使いたいというご要望がありました。

これは、一見簡単なことに思えます。しかし、1980 年代に決定された DNS プロトコルの設計思想が、これを見た目より困難にしているのです。DNS には、CNAME と呼ばれる機能があり、所有者はドメインの一部のホスティングを他のプロバイダーに任せられるようになっています。しかしこの機能は、ドメインのルートやトップレベルでは使えません。先に書いたような要望に答えるには、お客様のドメインを、当社が実際にホスティングしなければならないのです。当社でお客様ドメインをホスティングすると、Amazon S3、Amazon CloudFront、Elastic Load Balancing などに対し、その時点のいかなる IP アドレスのセットでも返す事が可能です。こういったサービスは拡張し続けており IP アドレスも追加され続けています。つまり、ユーザーの方ご自身で、ドメイン定義の中に容易にハードコードできるようなものではありません。

DNS ホスティングは小さなタスクではないのです。仮に、DNS に問題が発生すれば、ビジネス全体がオフラインになり得ます。しかし Amazon はこの需要を認識した後、いつも通りの姿勢で、つまり早急に、このことの解決に取り組みました。エンジニア達で小さなチームを編成し、その仕事を開始したのです。

DDOS 攻撃への対処

最大のチャレンジはなにかと尋ねたとき、どの DNS プロバイダーも、「distributed denial of service (DDOS) 攻撃」への対応である、と答えるでしょう。DNS は UDP プロトコルの一番上に

構築されています。つまり、DNS リクエストとは、この危険に満ちたインターネットの中で偽装されやすい存在なのです。一方で DNS は非常に重要なインフラストラクチャです。そういった事情から、誰かからビジネスを奪い取ったり、様々な理由からビジネスを停止させようとする「ブーター達」など不道徳な活動をする人達にとって、DNS は絶好の標的になります。また、間違った方向に導かれたため、まったくの個人的な行為が結果的に重大な罪につながっていることに気付いていない、やっかいな人達も時折登場します。その理由はともあれ、ドメインに対しては、毎日数千の DDOS 攻撃がしかけられているのです。

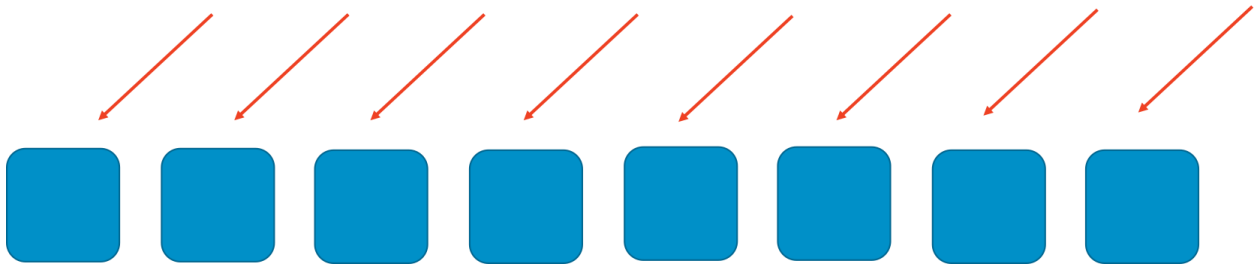
これらの攻撃の影響を軽減する 1 つのアプローチとしては、サーバーで巨大な容量を使用するというものがあります。ベースラインとして十分な容量を確保することは大切ですが、それでもこのアプローチでは、真に事態を打開できません。すべてのサーバーのプロバイダー達は数千ドルのコストを投入していますが、ボットネットを悪意であやつる攻撃者は、たった数ペニーで、さらに多くの疑似クライアントを生み出せるからです。したがって、サーバーに巨大なボリュームを追加していくのは、プロバイダーにとっては敗北の戦略でしかありません。

当社が Amazon Route 53 を作り上げた時点での最先端の DNS 防御は、トラフィックの「取り消し」を高速で行う各種のトリックを利用するような、ネットワークアプリケーションへの対応に特化したものでした。Amazon には、既存の社内 DNS サービスを行うため、多くの設備があります。当社は、ハードウェアベンダー達に、何か取れる対策がないか尋ねました。そこで明らかになったのは、すべての Route 53 ドメインをカバーするのに十分な設備を購入するには、数千万ドルの予算が必要となること、そして、その納入、設置、運用のために、スケジュールが数か月も追加されるということでした。これでは、この計画の緊急性や、予算抑制の努力などに適合しません。ということで、真剣に検討するには至りませんでした。当社が求めたのは、実際に攻撃を受けているドメインの防御だけにリソースを消費できる、なんらかの方法でした。そこで、必要は発明の母である、という古典的な法則に立ち戻ってみたのです。ここでの必要性とは、稼働率が 100 パーセントの世界規模の DNS サービスを、妥当な規模のリソースで実現するということです。そこで生まれた発明が、シャッフルシャーディングです。

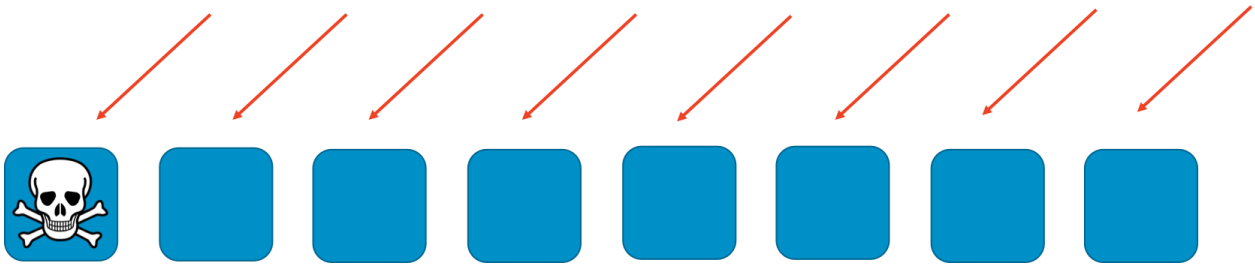
シャッフルシャーディングとは?

シャッフルシャーディングはシンプルかつパワフルです。当初想定していたものを超えるパワーがあります。この使用実績は多く、AWS が、各お客様にシングルテナントを提供するために使うマルチテナントサービスを、コスト効率よく実現できるようにする主要パターンとなりました。

シャッフルシャーディングの仕組みをご理解いただくために、まず、通常のシャーディングでのシステムに、どの程度のスケーラビリティと回復性を与えられるか考えてみます。たとえば、システムもしくはサービスが水平にスケーラブルで、8 人のワーカーで構成されているとします。次の画像が、ワーカー達と彼らからのリクエストを示しています。ワーカーとは、サーバーであったり、クエリ、もしくはデータベースなどのことで、それが「何」であったにせよ、システムを構成しています。



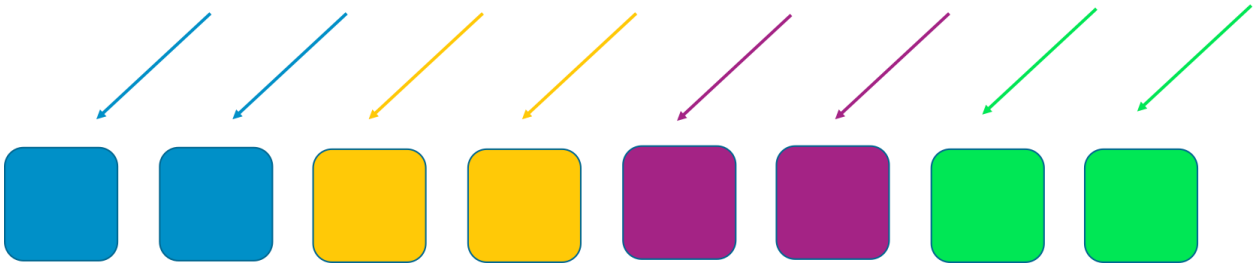
シャーディングなしの状態だと、ワーカーの一団がすべてを処理します。各ワーカーは、あらゆる種類のリクエストを処理できなければなりません。これは、効率や冗長性に関しては優れた手法です。1 つのワーカーがダウンしても、残りの 7 つのワーカーがその作業を吸収するので、システムの容量にはさほど厳しい要求は生じません。しかしながら、仮に特定のリクエストとか、DDOS 攻撃に見られるような大量のリクエストの流入により障害が引き起こされると、大きな問題が浮かび上がります。次の 2 つの画像に、そういった攻撃の進行過程を示します。



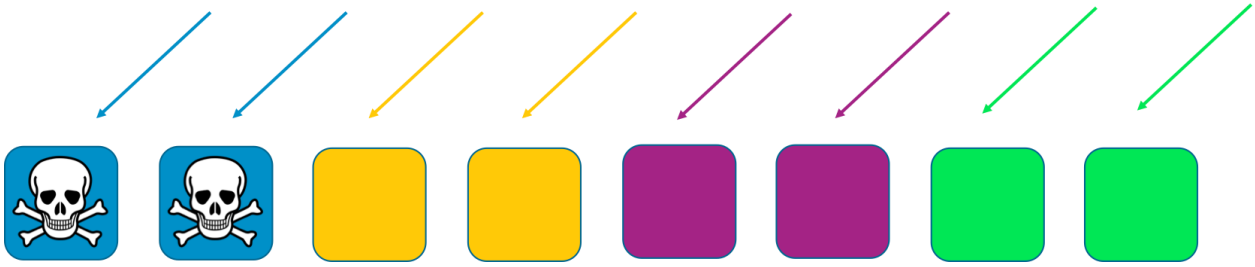
これによる障害は、最初に攻撃されたワーカーを停止させるだけでなく、処理を引き受けた残りのワーカーにも順次伝搬して行くのです。この障害は、あっというまに全ワーカーを無力化し、サービス全体を停止に追い込みます。

この種の障害が及ぼす範囲は「あらゆるモノとあらゆるヒト」におよびます。 全サービスがダウンします。顧客全員に影響がおよぶのです。可用性エンジニアリングの領域でいうところの、最適ではない状態です。

これには、標準のシャーディングで良い対応ができます。ワーカーの一団を 4 つのシャードに分割することで、効率性を削ぐ代りに影響範囲を狭められます。次の画像は、DDOS 攻撃の影響をシャーディングがどのように抑制するか説明しています。



この例では、各シャードには 2 ワーカーが属しています。シャード全体で、顧客のドメインなどのリソースを分割します。これでも冗長性は維持されますが、各シャードには 2 つのワーカーしか存在しないため、何かの問題が起きた時のために、システムにはゆとりのある容量が必要です。その見返りとして、攻撃の影響範囲は顕著に抑制されます。



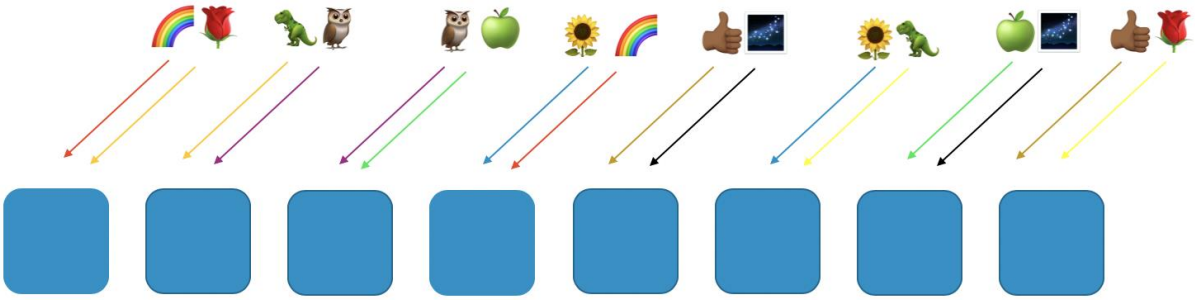
シャーディングを使う場合、影響の範囲はシャードの数に応じて小さくなります。この例での 4 シャードの場合、1 人のユーザーに何かの問題が発生しているということは、それをホストするシャード、そして、同じシャードの他の全ユーザーが影響をうけていると考えられます。しかしながら、このシャードが担当しているのは、サービス全体の 1/4 だけです。25 パーセントの影響なら、100 パーセントよりかなりましです。そして、シャッフルシャーディングを用いると、そのメリットを大幅に拡大できます。

シャッフルシャーディングでは、仮想シャードを作成し 2 ワーカーを割り当てます。そして、ユーザーやリソース、それ以外の分離したいあらゆるものを、この仮想シャードの一つに割り当てます。

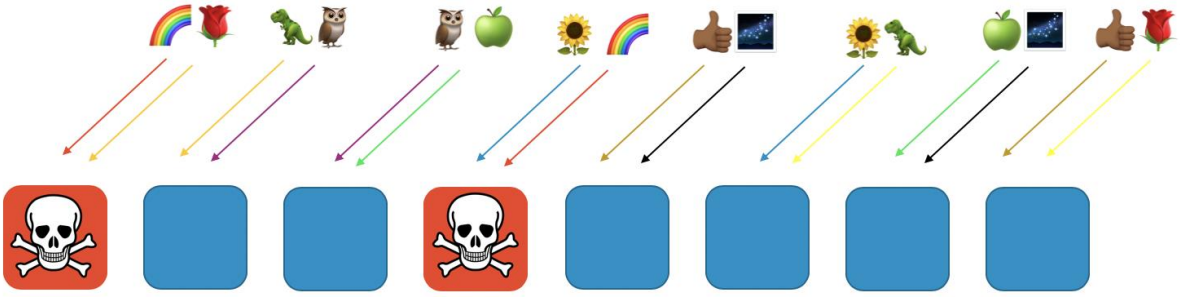
次の画像では、8 ワーカーと 8 ユーザーの場合で各シャードに 2 ワーカーを割り当てたときの、シャッフルシャーディングのレイアウト例を示しています。通常は、ワーカーよりユーザーの数が多くなりますが、規模を小さく考えたほうが追いかけやすいでしょう。ここでは、レインボウカスタマーとローズカスタマー、2 人のユーザーに注目します。

この例では、レインボウカスタマーは、第 1 のワーカーと第 4 のワーカーに割り当てられています。これら 2 つのワーカーにより、ユーザーに対するシャッフルシャーディングが構成されます。他のユーザーには、それぞれ違った仮想シャードと別々の 2 ワーカーの組合せが割り当

てられます。たとえば、ローズカスタマーにも第 1 のワーカーが割り当てられていますが、もう 1 つ割り当てられるのは第 8 のワーカーです。



1 つめと 4 つめのワーカーに割り当てられたレインボウユーザーに (有害なリクエストや大量のリクエストによる) 問題が発生した場合は、その仮想シャードにも障害が起きますが、他のシャッフルシャード全体にそれがおよぶことはありません。実際、最大でも、もう 1 つのシャッフルシャードでのワーカーが影響を受けるだけです。リクエスを送る側に耐障害性があり、この仕組みを (たとえばリトライにより) 避けることができたとしても、次の画像に示す通り、残りのシャードに割り当てられたユーザーやリソースは中断されることはありません。



別のケースとして、レインボウカスタマーにサービスを提供している全ワーカーに障害があったり、それが攻撃を受けたりしていても、他のワーカーに影響が及ぶことはあり得ません。レインボウとワーカーを共有しているローズカスタマーやサンフラワーカスタマーであっても、影響は受けません。次の画像にあるように、ローズカスタマーは第 8 ワーカーからサービスを提供され、サンフラワーカスタマーは第 6 ワーカーのサービスを受けられるからです。

障害発生の際には、サービス全体の 1/4 が失われる可能性は残りますが、このユーザーやリソースの配置方法により、シャッフルシャーディングでの影響範囲は顕著に抑制できるのです。8 ワーカーのケースでは、それぞれ 2 ワーカーによる 28 通りの一意の組み合わせが選べます。これは、28 のシャッフルシャードが存在することを意味します。ここで、数百以上のユーザーがおり、各ユーザーをシャッフルシャードに割り当てる場合には、障害が及ぼす影響の範囲は 1/28th になります標準的なシャーディングに比べ 7 倍の効果です。

そして実に嬉しいことに、ワーカーとユーザーがもっと多い場合には、この比率もずっと大きなものになります。こういった状況では、スケーリングに関する多くの課題が難しさを増すものですが、シャッフルシャーディングでは効率が改善するのです。実際のところ、十分なワーカーが存在すれば、より多くのシャッフルシャードを設定しユーザーを割り当てられ、各ユーザーを分離することができます。

Amazon Route 53 とシャッフルシャーディング

これは Amazon Route 53 においてどんな働きをするのでしょう? Route 53 では、全 2048 の仮想ネームサーバーに容量を割り当てています。Route 53 のホスティングをする物理サーバーと通信はしないので、これらは仮想のサーバーです。容量の管理をするために、これらを移動させることもできます。そして、各ユーザーのドメインは、4 つの仮想ネームサーバーでのシャッフルシャードに割り当てています。これらの数字から、なんと 7 億 3 千万のシャッフルシャードが設定できることになるのです。設定可能なシャッフルシャード数が多いので、各ドメインに対し一意のシャッフルシャードを割り当てることができます。実はさらに大型化すること

も可能で、それにより、ユーザーのドメインが他のユーザーのドメインと共有するネームサーバーは 2 つ以内になります。

これが生み出す結果は驚異的なものです。1 つのユーザードメインが DDOS 攻撃の標的になったときは、このドメインに割り当てられた 4 つのネームサーバーでトラフィックが急増しますが、その他のユーザードメインにはなにも起こりません。また、標的となり、良くない一日を過ごしているユーザーを見捨てた訳ではありません。シャッフルシャーディングでは、標的にされたユーザーを特定し、攻撃に対応するため特別なキャパシティーの中に隔離することができます加えて、トラフィック抑制のため、AWS Shield に当社専用のレイヤーも開発済みです。とはいえ、シャッフルシャーディングは、この主のイベントが発生している間であっても全 Route 53 ユーザーの操作性がシームレスになるよう、非常に大きな役割をはたすものです。

まとめ

当社では、他の多くのシステムにもシャッフルシャーディングを組み込んできました。加えて、再帰的シャッフルシャーディングなどの改良点も追加してきています。これは、複数のレイヤーでアイテムをシャードするもので、ユーザーの下にいるユーザーも分離できます。シャッフルシャーディングには優れた適応性があります。既存リソースをアレンジするには賢い手法です。また通常は追加コストなしで利用できるため、予算枠や経済性を考慮しても、とてもよい改善策の 1 つといえます。

シャッフルシャーディングの使用にご関心がある方は、Route 53 Infima のライブラリーで、当社のオープンソースをチェックしてください。このライブラリーには、リソースの割り当てやアレンジに使える、いくつかの異なるシャッフルシャーディングの実装が収められています。