셔플 샤딩을 사용한 워크로드 격리

Colm MacCárthaigh

셔플 샤딩을 사용한 워크로드 격리

오늘날 Amazon Route 53 는 전 세계의 수많은 대기업과 유명 웹 사이트를 호스팅하고 있지만 그 시작은 초라했습니다.

DNS 호스팅의 시작

AWS 가 서비스를 제공하기 시작한 지 얼마 되지 않아 AWS 고객들은 자사 도메인의 “루트”에서 당사의 Amazon Simple Storage Service(S3), Amazon CloudFront 및 Elastic Load Balancing 서비스를 이용할 수 있기를 희망한다는 점을 분명히 했습니다. 즉, "www.amazon.com" 같은 이름뿐만 아니라 "amazon.com" 같은 이름에도 사용하게 해 달라는 말입니다.

이는 매우 간단하게 보입니다. 하지만 1980 년대에 이루어진 DNS 프로토콜 설계상의 결정 때문에 보기보다 쉽지 않습니다. DNS 에는 도메인 소유자가 도메인의 일부를 다른 제공자에게 오프로드하여 호스팅할 수 있도록 하는 CNAME 이라는 기능이 있지만, 도메인의 루트 또는 최상위 수준에서는 작동하지 않습니다. 고객의 요구에 부응하려면 고객의 도메인을 실제로 호스팅해야 합니다. 고객의 도메인을 호스팅할 경우 Amazon S3, Amazon CloudFront 또는 Elastic Load Balancing 의 현재 IP 주소 세트를 무엇이든 반환할 수 있습니다. 이들 서비스는 IP 주소를 지속적으로 확장 및 추가하므로 고객이 도메인 구성에 쉽게 하드 코딩할 수 있는 것이 아닙니다.

따라서 DNS 를 호스팅하는 것은 간단한 작업이 아닙니다. DNS 에서 문제가 발생하면 전체 비즈니스가 오프라인 상태가 됩니다. 하지만 고객의 요구를 파악한 후 Amazon 은 늘 그렇듯 이 문제를 시급히 해결하기로 작정했습니다. 소수의 엔지니어로 팀을 꾸리고 바로 작업에 착수했습니다.

DDoS 공격 대응

DNS 공급자에게 가장 큰 문제가 무엇인지 물으면 하나같이 DDoS(분산 서비스 거부) 공격에 대응하는 것이라고 말할 것입니다. DNS 는 UDP 프로토콜을 기반으로 하기 때문에 험한 인터넷 환경의 어디서든 DNS 요청이 스푸핑될 수 있습니다. DNS 도 중요한 인프라이므로, 이 두 가지 문제가 만나면 기업을 갈취하려는 비양심적인 범죄자들, 다양한 이유로 가동 중단을 촉발하는 것을 목표로 삼는 "부터들", 그리고 가끔 그들이 실제 개인적으로 처벌받을 수 있는 심각한 범죄를 저지르고 있다는 사실을 깨닫지 못하는 길잃은 골칫거리 양들에게 DNS 가 좋은 표적이 됩니다. 이유를 막론하고 매일 수천 건의 DDoS 공격이 도메인을 표적으로 자행되고 있습니다.

이 같은 공격을 완화하는 방법 중 하나는 대규모 서버 용량을 사용하는 것입니다. 용량의 기준을 효과적으로 정하는 것은 물론 중요하지만, 이 방법은 사실 확장성이 떨어집니다. 공급자가 서버를

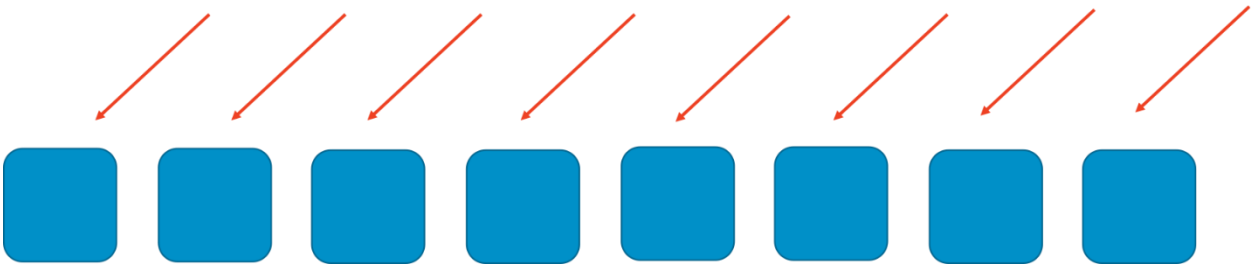
추가할 때마다 수천 달러의 비용이 발생하지만, 공격자들은 해킹한 봇넷을 사용해 불과 몇 센트의 비용으로 가짜 클라이언트를 추가할 수 있습니다. 공급자의 입장에서 막대한 서버 용량을 추가하는 방법은 패배가 뻔히 보이는 전략입니다.

Amazon Route 53 를 구축할 당시 DNS 방어 기술은 다양한 기법을 이용하여 매우 빠른 속도로 트래픽을 "스크러빙"하는 전문화된 네트워크 어플라이언스의 형태였습니다. Amazon 의 기존 사내 DNS 서비스에는 이 같은 어플라이언스가 이미 많이 사용되고 있었기에 당시 하드웨어 공급자에게 적용 가능한 다른 솔루션은 없는지 물었습니다. 이 어플라이언스로 모든 Route 53 도메인을 하나하나 완벽하게 지원하려면 수천만 달러의 비용이 소요되고, 어플라이언스를 배송받아 설치하고 운영하기 위해서는 수개월의 일정을 따로 잡아야 한다는 사실을 알게 되었습니다. 시급을 다투는 Amazon 의 계획이나 비용 절감이라는 측면에서도 이 솔루션은 현실성이 없었기에 중요하게 고려하지 않았습니다. 따라서 실제로 공격이 발생하는 도메인을 방어하는 데만 리소스를 투입할 방법을 찾아야 합니다. 고심 끝에 필요는 발명의 어머니라는 오랜 원칙을 따르기로 했습니다. 적절한 양의 리소스를 사용하여 100% 가동률을 보장하는 세계적인 수준의 DNS 서비스를 빠르게 구축해야 한다는 것이 우리의 "필요"였습니다. 그리고 그 필요가 낳은 발명은 바로 셔플 샤딩이었습니다.

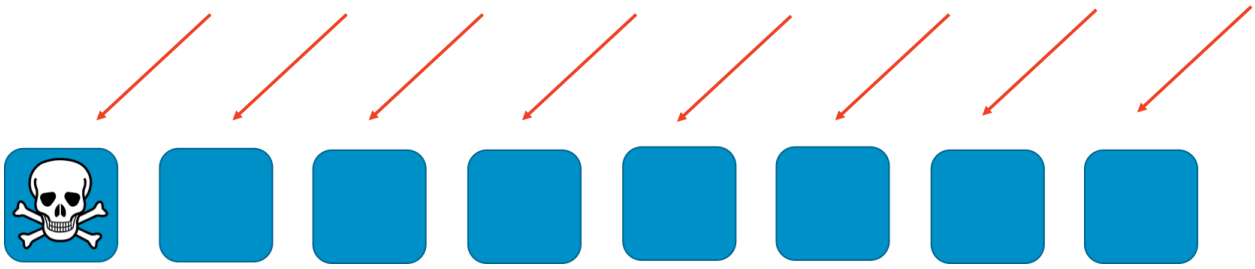
셔플 샤딩이란 대체 무엇일까요?

셔플 샤딩은 단순하지만 강력한 기술입니다. 저희가 처음에 생각한 것보다도 더 강력했습니다. 반복해서 사용하면서, 이 기술은 AWS 가 각 고객에게 단일 테넌트 환경을 지원하는 비용 효율적인 멀티 테넌트 서비스를 제공하는 데 있어서 핵심적인 패턴이 되었습니다.

셔플 샤딩의 작동 원리를 이해하려면 먼저 일반적인 샤딩을 통해 시스템의 확장성과 복원력을 확보하는 방법부터 살펴보아야 합니다. 8 개의 워커로 이루어진 수평 확장이 가능한 시스템이나 서비스를 예로 들어보겠습니다. 다음 이미지는 워커와 그 요청을 보여줍니다. 워커는 서버일 수도 있고, 대기열일 수도 있으며, 데이터베이스일 수도 있습니다. 시스템을 구성하는 "것"이라면 무엇이든 워커가 될 수 있는 겁니다.



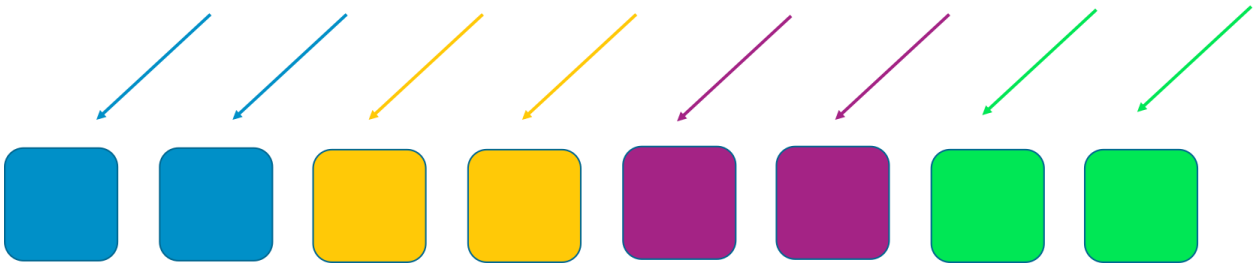
샤딩 기술을 적용하지 않으면 워커의 플릿에서 모든 작업이 처리됩니다. 각 워커는 어떤 요청이든 처리할 수 있어야 합니다. 이는 효율성과 이중화라는 측면에서는 바람직합니다. 워커 중 하나에서 장애가 발생하면 다른 7 개의 워커가 작업을 처리할 수 있기 때문에 시스템에 필요한 유휴 용량이 비교적 작습니다. 하지만 특정한 종류의 요청으로 인해 또는 DDoS 공격을 받는 경우처럼 요청이 급증하여 장애가 발생할 경우 큰 문제에 직면하게 됩니다. 다음 2 개의 이미지는 이 같은 공격의 진행 과정을 보여줍니다.



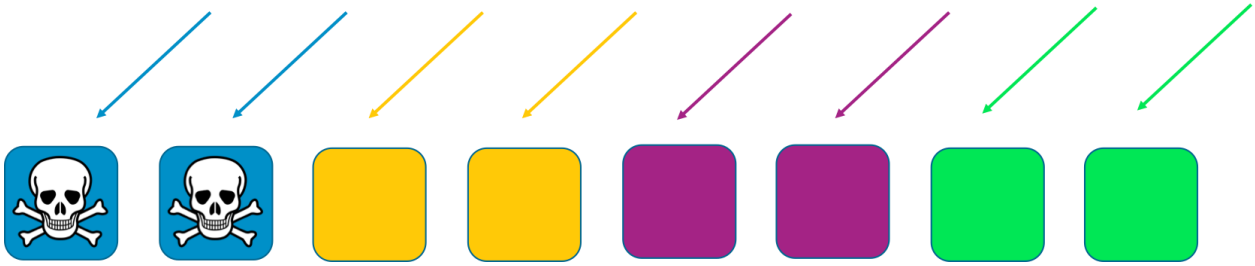
문제가 발생하면서 영향을 받은 첫 번째 워커가 작동하지 않게 되고, 곧 그 역할을 이어받은 나머지 워커도 연쇄적으로 작동 불능 상태가 됩니다. 이 문제는 모든 워커를 순식간에 작동 불능 상태로 만들고 결국 전체 서비스가 중단될 수 있습니다.

이 같은 장애의 피해 범위는 “모든 구성 요소와 모든 사람”입니다. 전체 서비스가 가동 중단됩니다. 모든 고객이 피해를 입습니다. 가용성 엔지니어링에서는 “최적이란 없다”라는 말이 있습니다.

일반적인 샤딩을 적용하면 상황을 개선할 수 있습니다. 즉, 플릿을 4 개의 워커 샤드로 나누면 효율성을 조금 떨어뜨리는 대신 피해의 범위를 줄일 수 있습니다. 다음 2 개의 이미지는 샤딩을 통해 DDoS 공격의 피해가 어떻게 억제되는지 보여줍니다.



이 예에서 각 샤드는 2 개의 워커로 이루어져 있습니다. 고객 도메인과 같은 리소스를 여러 샤드에 걸쳐 분할합니다. 여전히 이중화된 상태지만 샤드당 워커가 2 개뿐이므로 시스템에서 장애에 대응하기 위한 유휴 용량을 더 많이 확보해야 합니다. 그 대가로 피해의 범위는 크게 축소됩니다.

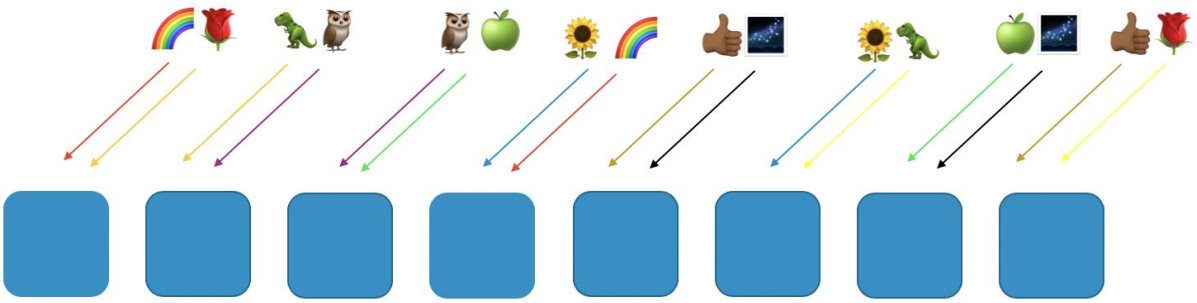


이 샤딩 환경에서 피해의 범위는 샤드의 수에 비례하여 줄어듭니다. 샤드가 4 개인 이 예에서 고객 환경에 문제가 발생하면 해당 환경을 호스팅하는 샤드가 영향을 받게 되고 해당 샤드상의 다른 고객도 모두 영향을 받습니다. 하지만 해당 샤드는 전체 서비스의 1/4 에 해당합니다. 25%의 피해는 100% 피해보다 훨씬 낫습니다. 셔플 샤딩을 적용하면 이보다도 훨씬 더 상황을 개선할 수 있습니다.

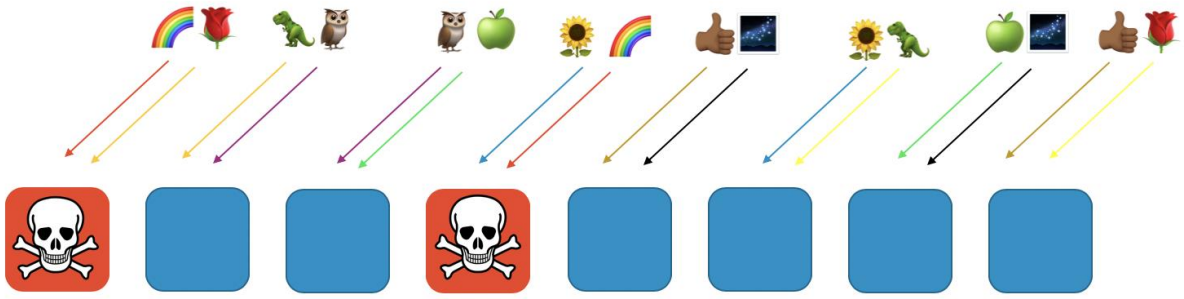
셔플 샤딩의 경우 각각 2 개의 워커로 이루어진 가상 샤드를 만들어, 고객이나 리소스 또는 그 외에 격리해야 할 것이라면 무엇이든 해당 가상 샤드 중 하나에 할당합니다.

다음 이미지는 8 개의 워커와 각각 2 개의 워커에 할당된 8 개의 고객으로 이루어진 셔플 샤딩 레이아웃의 예를 보여줍니다. 통상적으로 워커보다 고객 수가 훨씬 많지만, 여기서는 이해하기 쉽도록 작은 규모로 예를 들겠습니다. 여기서 무지개 고객과 장미 고객, 이 2 개의 고객에 주목해보겠습니다.

이 예에서는 무지개 고객을 첫 번째 워커와 네 번째 워커에 할당합니다. 이 두 워커의 조합은 해당 고객의 셔플링된 샤드를 구성합니다. 다른 고객은 워커 2 개의 고유한 조합으로 다른 가상 샤드에 할당됩니다. 예를 들어 로즈 고객도 첫 번째 워커에 할당되지만, 나머지 하나의 워커는 8 번째 워커입니다.



1 번 워커와 4 번 워커에 할당된 무지개 고객에게서 문제가 발생하면(예: 악의적인 요청 또는 요청의 급증) 그 문제는 해당 가상 샤드에 영향을 미치지만 다른 셔플 샤드에는 온전히 영향을 미치지 않습니다. 사실 다른 셔플 샤드의 워커 중 하나는 대부분 영향을 받습니다. 요청자가 내결함성을 갖추고 있고 (재시도 등의 방법으로) 이 문제를 해결할 수 있다면, 다음 이미지와 같이 나머지 샤드에서 고객 또는 리소스에 중단 없이 서비스를 계속 제공할 수 있습니다.



바꿔말하면, 무지개 고객을 서비스하는 모든 워커에서 문제가 발생하거나 공격을 당하더라도 다른 워커는 전혀 영향을 받지 않습니다. 각각 무지개 고객과 워커 중 하나를 공유하는 장미 고객과 해바라기 고객을 비롯해 다른 고객들은 영향을 받지 않습니다. 다음 이미지에서 보듯 장미 고객은 8 번 워커에서 서비스를 제공받을 수 있고, 해바라기 고객은 6 번 워커에서 서비스를 제공받을 수 있습니다.

셔플 샤딩의 경우에도 문제가 발생하면 서비스의 1/4 이 손상되지만 고객 또는 리소스를 할당하는 방식 덕분에 피해의 범위는 훨씬 더 축소됩니다. 8 개의 워커를 사용할 경우, 2 개의 워커로 이루어진 28 가지 고유한 조합을 만들 수 있으므로 셔플 샤드의 수가 28 개인 셈입니다. 고객이 수백 개 이상이고 고객마다 셔플 샤드를 할당한다면 문제로 인한 피해의 범위는 1/28 에 불과합니다. 일반 샤딩보다 7 배 더 개선된 결과죠.

워커 수와 고객 수가 늘어날수록 효과가 더 커진다는 점이 특히 고무적입니다. 대부분의 확장 문제는 규모가 늘어날수록 해결하기가 더 어려워지지만 셔플 샤딩은 오히려 더 효과가 높아집니다. 사실 워커 수만 충분하다면 고객 수보다 더 많은 셔플 샤드를 만들어 고객을 각각 격리할 수 있습니다.

Amazon Route 53와 셔플 샤딩

이러한 것들이 Amazon Route 53 에는 어떤 점에서 유용할까요? 저희는 Route 53 의 용량을 총 2,048 개의 가상 이름 서버로 재구성하기로 결정했습니다. 이들 서버는 Route 53 를 호스팅하는 물리적 서버에 일치하지 않기 때문에 가상 서버라고 부릅니다. 가상 서버는 용량 관리가 용이하도록 이동할 수 있습니다. 그리고 모든 고객 도메인을 4 개의 가상 이름 서버로 이루어진 셔플 샤드에 할당합니다. 이 같은 숫자로 계산해보면 가능한 셔플 샤드 수가 무려 7,300 억 개에 달합니다. 모든 도메인에 고유한 셔플 샤드를 할당할 만큼 충분한 셔플 샤드가 확보되는 것입니다. 사실 규모를 더욱 늘려 고객 도메인 간에 3 개 이상의 가상 이름 서버가 절대 공유되지 않도록 보장할 수도 있습니다.

결과는 놀랍습니다. 특정 고객 도메인이 DDoS 공격의 표적이 되면 해당 도메인에 할당된 4 개의 가상 이름 서버에서 트래픽이 급증하지만 다른 고객의 도메인에서는 아무 문제도 나타나지 않습니다. 그렇다고 표적이 된 고객을 그냥 두는 것은 아닙니다. 셔플 샤딩에서는 표적이 된 고객을 식별한 후 특별한 전용 공격 용량을 사용해 격리할 수 있습니다. 또한 Amazon 은 AWS Shield 트래픽 스크러버로 이루어진 고유한 계층을 개발했습니다. 하지만 어쨌든 셔플 샤딩은 이 같은 이벤트가 발생하는 경우에도 원활한 전체 Route 53 고객 경험을 보장하는 데 있어 큰 차이를 만들어냈습니다.

결론

셔플 샤딩은 Amazon 의 다른 시스템에도 많이 적용되었습니다. 또한 여러 계층에 걸쳐 샤드를 구성하여 고객의 고객까지 격리하는 반복적 셔플 샤딩을 비롯해 여러 가지 기술적 발전도 있었습니다. 셔플 샤딩은 광범위하게 적용할 수 있는 기술이며, 기존 리소스를 스마트하게 재구성할 수 있는 기술입니다. 또한 일반적으로 추가 비용이 발생하지 않기 때문에 비용 절감과 경제성에 있어서도 탁월합니다.

셔플 샤딩을 직접 사용해보고 싶으신 분은 오픈 소스 Route 53 Infima 라이브러리를 참조하시기 바랍니다. 이 라이브러리에는 리소스를 할당하거나 재구성하는 데 활용할 수 있는 셔플 샤딩의 다양한 구현 예가 수록되어 있습니다.