Isolamento del carico di lavoro utilizzando lo sharding casuale

Colm MacCárthaigh

Isolamento del carico di lavoro utilizzando lo sharding casuale

Oggi, Amazon Route 53 offre un servizio di hosting a molte delle più grandi aziende al mondo e ai siti web più popolari, ma i suoi inizi sono molto più umili.

La sfida al DNS hosting

Non molto tempo dopo che AWS aveva iniziato a offrire i propri servizi, i clienti di AWS fecero capire che desideravano essere in grado di usare i servizi Amazon Simple Storage Service (S3), Amazon CloudFront ed Elastic Load Balancing alla “radice” del loro dominio, ossia non solo per nomi come “www.amazon.com” ma anche per nomi come “amazon.com”.

Ciò può sembrare molto semplice. Tuttavia, a causa di una decisione di progettazione del protocollo DNS presa negli anni Ottanta, è più difficile di quanto non sembri. DNS ha una funzione, denominata CNAME, che consente al proprietario di un dominio di trasferirne parte a un altro provider che vuole offrire il servizio di hosting, ma che non è operante alla radice ovvero al livello superiore di un dominio. Per rispondere alle esigenze dei nostri clienti, avremmo dovuto effettivamente ospitare i loro domini. Quando forniamo il servizio di hosting al dominio di un cliente, possiamo restituire il set corrente di indirizzi IP, qualunque esso sia, relativo ad Amazon S3, Amazon CloudFront o Elastic Load Balancing. Questi servizi si espandono e aggiungono continuamente indirizzi IP, per cui non si tratta di una funzione che i clienti potrebbero impostare facilmente come hardcoded nelle configurazioni dei loro domini.

Non è semplice attuare l’hosting DNS. Se DNS riscontra problemi, un intero business può essere offline. Ma dopo avere identificato l’esigenza, ci siamo impegnati a soddisfarla nel modo tipico di Amazon – urgentemente. Abbiamo incaricato un team di ingegneri e ci siamo messi al lavoro.

Gestione degli attacchi di tipo Distributed Denial of Service (DDoS)

Se si chiede a qualunque DNS provider quale sia la sfida più complessa alla quale deve far fronte, la risposta sarà: come gestire gli attacchi di tipo Distributed Denial of Service (DDoS). DNS viene realizzato a partire dal protocollo UDP, il che significa che le richieste DNS possono essere soggette a spoofing da parte dei pirati informatici. Poiché DNS è anche un’infrastruttura critica, questa combinazione ne fa un bersaglio attraente per attori senza scrupoli che cercano di effettuare estorsioni ai danni delle aziende, “booter” che mirano a causare interruzioni del servizio per una varietà di motivi e l’accidentale persona maldestra che sembra non capire che sta commettendo un reato grave con vere conseguenze personali. Indipendentemente dal motivo, ogni giorno vengono sferrati migliaia di attacchi DDOS contro i domini.

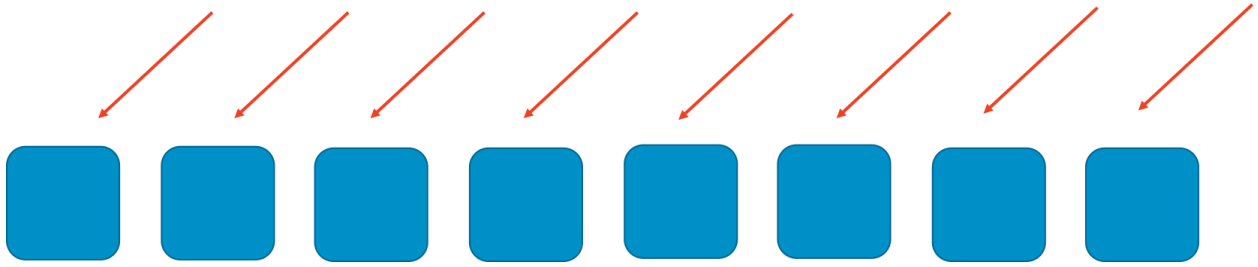
Un approccio alla mitigazione di questi attacchi consiste nell’usare enormi volumi di capacità del server. Sebbene sia importante avere una base adeguata di capacità, questo approccio non è veramente scalabile. Ogni server aggiunto da un provider costa migliaia di dollari, ma gli utenti malintenzionati posso aggiungere ulteriori clienti falsi per centesimi se stanno utilizzando botnet compromessi. Per i provider, aggiungere enormi volumi di capacità del server è una strategia perdente.

Quando abbiamo realizzato Amazon Route 53, lo stato dell'arte per quanto riguarda la difesa DNS consisteva nell'uso di appliance di rete che potevano usare varie strategie per "pulire" il traffico a una frequenza molto alta. Avevamo molte di queste appliance in Amazon per i nostri servizi DNS interni esistenti, e abbiamo chiesto ai fornitori di hardware cos'altro era disponibile. Abbiamo scoperto che l'acquisto di un numero sufficiente di appliance per coprire completamente ogni singolo dominio Route 53 avrebbe comportato una spesa di decine di milioni di dollari e aggiunto mesi al nostro programma per consegnarle, installarle e metterle in servizio. Ciò non si adattava all'urgenza dei nostri piani né ai nostri sforzi volti a contenere i costi, per cui non abbiamo mai considerato seriamente questa strategia. Avevamo bisogno di trovare un modo per spendere risorse solo per difendere domini che sono effettivamente soggetti ad attacchi. Abbiamo fatto ricorso al principio che la necessità è la madre di ogni invenzione. La nostra necessità era quella di creare rapidamente un servizio DNS avanzato con uptime del 100% impiegando una quantità limitata di risorse. La nostra invenzione: lo sharding casuale.

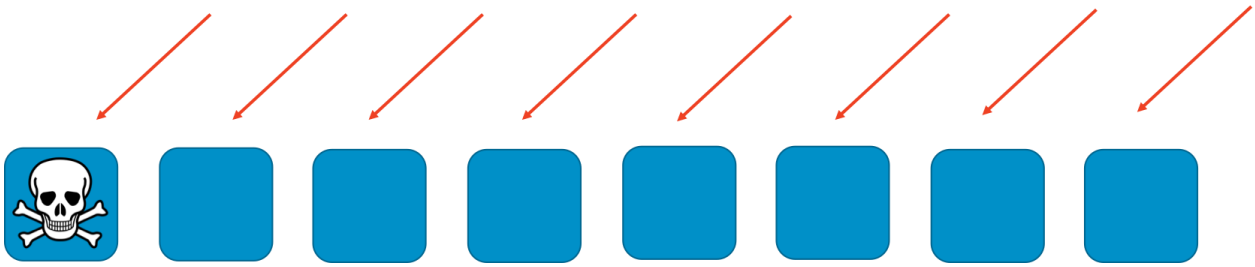
Cos'è lo sharding casuale?

È una soluzione semplice ma potente. Anche più potente di quanto inizialmente non ci fosse sembrato. L'abbiamo usata ripetutamente ed è diventata uno schema essenziale che consente ad AWS di offrire economici servizi multi-tenant che assicurano a ciascun cliente un'esperienza single-tenant.

Per capire il principio di funzionamento dello sharding casuale, consideriamo anzitutto come un sistema possa essere reso più scalabile e resiliente tramite lo sharding ordinario. Si immagini un servizio o un sistema scalabile in orizzontale composto da otto elementi. La seguente immagine illustra gli elementi e le loro richieste. Gli elementi potrebbero essere server, code o database, quali che siano i componenti del sistema.



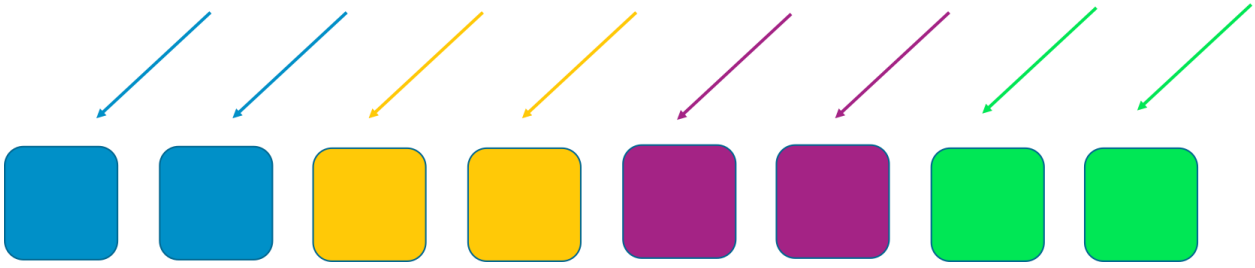
In assenza di sharding, il gruppo di elementi gestisce tutto il carico di lavoro. Ciascuno di essi deve essere in grado di gestire qualunque richiesta e ciò aumenta notevolmente l'efficienza e la ridondanza. Se un elemento non funziona, gli altri sette possono assorbire il carico di lavoro, per cui il sistema ha bisogno di una capacità inutilizzata relativamente piccola. Tuttavia, nasce un problema notevole se i guasti possono essere causati da un particolare tipo di richiesta oppure da un'ondata di richieste, come un attacco DDOS. Le due immagini seguenti mostrano la progressione di un tale attacco.



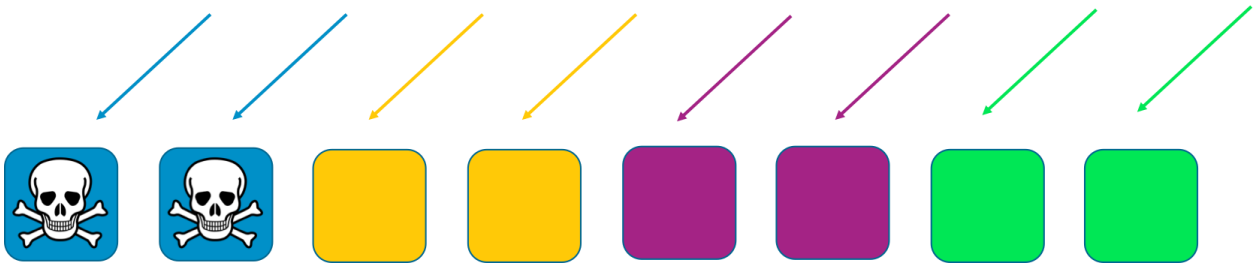
Il problema eliminerà il primo elemento attaccatto, ma poi procederà in sequenza attraverso gli altri elementi mentre questi si sostituiscono all'elemento eliminato. Il problema può eliminare velocemente tutti gli elementi e interrompere l'intero servizio.

L'ambito dell'impatto di questo tipo di guasto è "tutto e chiunque". L'intero servizio si arresta e ogni cliente ne risente gli effetti. Come si dice nell'ingegneria della disponibilità: non è ottimale.

Con un normale sharding possiamo fare di meglio. Se dividiamo il gruppo in 4 sottogruppi di elementi, possiamo raggiungere un compromesso fra efficienza e ambito dell'impatto. Le seguenti due immagini mostrano come lo sharding possa limitare l'impatto di un attacco DDOS.



In questo esempio, ciascun sottogruppo consiste di due elementi. Dividiamo risorse, come i domini di un cliente, fra i sottogruppi. Abbiamo ancora ridondanza, ma poiché vi sono solo due elementi per sottogruppo, dobbiamo mantenere più capacità inutilizzata nel sistema per gestire eventuali guasti. In compenso, l'ambito dell'impatto viene ridotto considerevolmente.

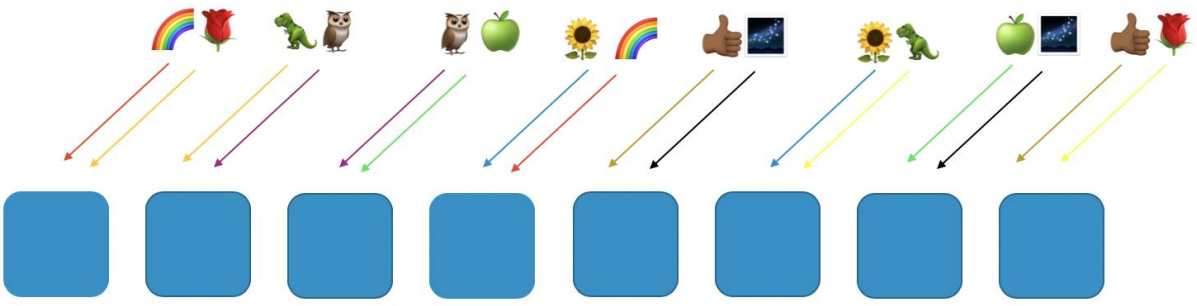


In questa configurazione di sharding, l'ambito dell'impatto è ridotto dal numero di sottogruppi. In questo caso, con quattro sottogruppi, se un cliente riscontra un problema, il sottogruppo che ospita il cliente potrebbe risentirne l'effetto, così come tutti gli altri clienti ospitati in quel sottogruppo. Tuttavia, tale sottogruppo rappresenta solo un quarto del servizio complessivo. Un impatto del 25% è molto migliore di un impatto del 100%. Con lo sharding casuale possiamo fare ancora meglio in misura esponenziale.

Con lo sharding casuale creiamo sottogruppi virtuali di due elementi ciascuno e assegniamo i clienti o le risorse o qualunque altra cosa che desideriamo isolare, a uno di questi sottogruppi virtuali.

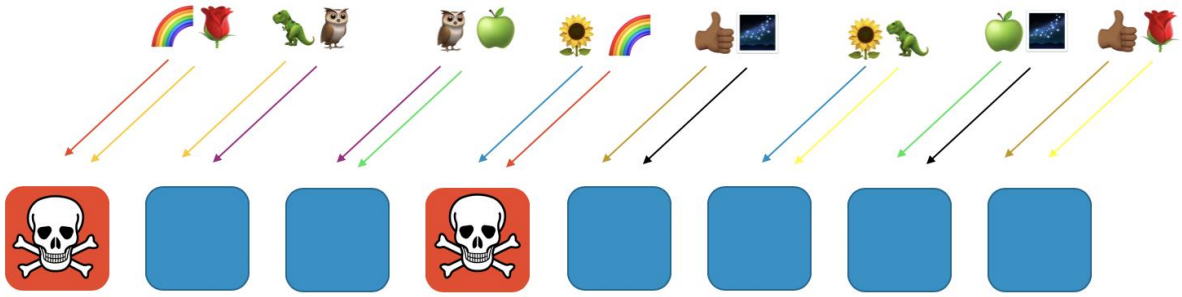
La seguente immagine mostra un esempio di disposizione di sharding casuale con otto elementi e otto clienti, ciascuno dei quali è assegnato a due elementi. Normalmente avremmo molti più clienti che elementi, ma è più facile seguire il ragionamento se manteniamo i numeri limitati. Focalizzeremo l'attenzione su due clienti – quello identificato dal simbolo dell'arcobaleno e quello identificato dal simbolo della rosa.

Nel nostro esempio, assegniamo il cliente "arcobaleno" al primo e al quarto elemento. La combinazione di questi due elementi costituisce il sottogruppo casuale di questo cliente. Altri clienti saranno assegnati a sottogruppi causali diversi, ciascuno con la propria combinazione di due elementi. Per esempio, il cliente "rosa" viene pure assegnato al primo elemento, ma il suo altro elemento è l'ottavo.



Se il cliente "arcobaleno" assegnato al primo e al quarto elemento ha un problema (come una richiesta pericolosa o un'ondata di richieste), il problema avrà effetto su quel sottogruppo virtuale ma non avrà un impatto completo su nessun altro sottogruppo casuale. Infatti, al massimo sarà interessato solo uno degli altri elementi del sottogruppo casuale. Se i richiedenti

sono a tolleranza di errore e possono aggirare il problema (per esempio, con tentativi successivi), il servizio può continuare ininterrotto per i clienti o le risorse sui sottogruppi rimanenti, come mostra la seguente immagine.



Ossia, mentre tutti gli elementi che servono il cliente “arcobaleno” potrebbero essere soggetti a un problema o un attacco, gli altri elementi non sono per niente coinvolti. Per quanto riguarda i clienti, ciò significa che anche se il cliente “rosa” e quello “girasole” condividono ciascuno un elemento con il cliente “arcobaleno”, non risentono alcun effetto. Il cliente “rosa” può ottenere il servizio dall’ottavo elemento e quello “girasole” dal sesto elemento, come mostrato nella seguente immagine.

Quando si verifica un problema, possiamo ancora perdere un quarto dell’intero servizio, ma il modo in cui i clienti o le risorse sono assegnati fa sì che l’ambito dell’impatto con lo sharding casuale sia considerevolmente ridotto. Con otto elementi, vi sono 28 combinazioni uniche di due elementi, il che significa che sono possibili 28 sottogruppi casuali. Se abbiamo centinaia o più clienti e assegniamo ciascuno di essi a un sottogruppo casuale, l’ambito dell’impatto dovuto a un problema è pari ad appena 1/28. Ossia 7 volte migliore rispetto al normale sharding.

È molto interessante vedere che i numeri diventano esponenzialmente migliori quanti più elementi e clienti si hanno. Molti problemi di scalabilità diventano più difficili con tali ordini di grandezza, ma lo sharding casuale diventa più efficace. Infatti, con un numero sufficiente di elementi, il numero dei sottogruppi casuali può essere superiore a quello dei clienti e ciascun cliente può essere isolato.

Amazon Route 53 e lo sharding casuale

Tutto ciò come aiuta Amazon Route 53? Con Route 53, abbiamo deciso di disporre la nostra capacità su un totale di 2048 server dei nomi virtuali. Questi server sono virtuali perché non corrispondono ai server fisici che ospitano Route 53. Possiamo spostarli per gestire più facilmente la capacità. Assegniamo quindi il dominio di ogni cliente a un sottogruppo casuale di quattro server dei nomi virtuali. Con queste cifre, il numero di possibili sottogruppi casuali è pari allo sbalorditivo valore di 730 miliardi. Abbiamo così tanti possibili sottogruppi casuali che possiamo assegnare un sottogruppo casuale unico a ogni dominio. In effetti, possiamo andare oltre e far sì che nessun dominio di alcun cliente condivida mai più di due server dei nomi virtuali con qualsiasi altro dominio di altri clienti.

I risultati sono straordinari. Se il dominio di un cliente è il bersaglio di un attacco DDOS, i quattro server dei nomi virtuali assegnati a tale dominio registreranno un notevole aumento transitorio del traffico che però non sarà osservato da nessun altro dominio di altri clienti. Ma non siamo rassegnati al fatto che il cliente preso di mira possa avere problemi. Lo sharding casuale ci permette di identificare e isolare il cliente preso di mira con una speciale capacità dedicata per la difesa da attacchi. Inoltre, abbiamo sviluppato il nostro livello proprietario di strumenti di pulitura (“scrubber”) del traffico AWS Shield. Tuttavia, lo sharding casuale comporta un’enorme differenza nell’assicurare che l’esperienza del cliente Route 53 complessiva sia della massima qualità anche mentre sono in corso questi eventi.

Conclusione

Abbiamo integrato lo sharding casuale in molti dei nostri altri sistemi. Inoltre, abbiamo ideato perfezionamenti, come lo sharding casuale ricorsivo, in cui creiamo sottogruppi su più livelli, isolando così il cliente di un cliente. Lo sharding casuale è enormemente adattabile. Rappresenta un modo ingegnoso per disporre risorse esistenti. Inoltre in genere non comporta costi aggiuntivi, per cui rappresenta un notevole miglioramento a cui possono dare impulso la frugalità e l’economicità.

Chi è interessato a usare autonomamente lo sharding casuale può consultare la nostra libreria open source Route 53 Infima, che include numerose implementazioni diverse dello sharding casuale utilizzabili per assegnare o disporre risorse.