[ENGLISH]

Hello Cloud Gurus, and welcome to this lecture.

And this lecture is all about the ProvisionedThroughputExceededException

and exponential backoff. So let's get started.

Now the provision to throughput exceeded exception is one that you might see on

your DynamoDB table if your request rate is too high for the

read and write capacity provisioned on the DynamoDB table.

So for example,

if your application is making too many requests into the DynamoDB

table,

so too many read and write requests for the table to handle based on its

read and write capacity provision. Now,

if you're using the AWS SDK,

it's going to automatically retry the requests until they are successful.

However, if you're not using the SDK,

you're going to have to configure your application to do one or both of the

following things. So firstly, you can reduce the request frequency,

so reduce the number of concurrent requests that are being sent to the DynamoDB

table, reduce the frequency of the requests that are being sent to the table,

and the second thing you can do is to implement exponential backoff.

Now what is exponential backoff? Well,

many components in a network or in a given system can generate errors

due to being overloaded by too many requests,

all coming in at the same time.

And that could be any component that you can think of,

so maybe networks switches, DNS servers, load balancers,

et cetera.

Pretty much any part of the network can experience overloading due to

too many concurrent requests.

And the usual way of dealing with this is to implement retries from the client

application. So for example,

for the client to keep on trying to send a request until it gets a response.

And all of the AWS SDKs, they all implement automatic retries,

which simply send the request again until it is successful.

But in addition to these simple retries,

all AWS SDKs also use exponential backoff.

And with exponential backoff,

this means that the requester uses progressively longer waits between

consecutive retries for improved flow control.

For example, after the first failed request, it might wait,

for example, 50 milliseconds before trying again.

And then if that retry fails,

it then might wait 100 milliseconds until trying again.

And then if that retry fails,

it might wait 200 milliseconds before trying again.

So it will continue like that until the request is successful.

And this gives significantly improved flow control in the hope that at some

point, the traffic is going to start flowing freely again,

and requests will start to be fulfilled again.

But if after about 1 minute this doesn't work,

it might be instead that your request size is actually exceeding the

throughput for your read and write capacity on the DynamoDB table.

So in this case, it may be worth investigating your provisioned throughput.

And if your workload is mainly Get requests,

you might be able to improve performance using either DynamoDB Accelerator,

or DAX, or using ElastiCache. However,

if it's mainly write requests that are causing the issue,

then maybe take a look at increasing your write capacity for the DynamoDB table.

So what are our exam tips? Well,

if you see the ProvisionedThroughputExceeded error,

this means that the number of requests into your DynamoDB table is too high.

Exponential backoff can be used to improve the flow control by retrying the

requests using progressively longer waits. However,

if your wait is getting up to around 1 minute,

it's definitely worth checking your read and write capacity units' settings and

seeing if they need to be adjusted. And just remember that exponential backoff,

it doesn't only apply to DynamoDB.

It's actually a feature of every single AWS SDK,

and it applies to many services within AWS, for example,

S3 buckets, if you're using CloudFormation heavily, and SES, et cetera.

And if you're not using the AWS SDK,

then you're going to need to handle this yourself in your application settings

or in your application code.

So the way to do that is just to reduce the frequency of your requests and

implement exponential backoff yourself. So that is the end of this lecture.

If you have any questions, please let me know if not,

feel free to move on to the next lecture. Thank you.

[SPANISH]

Hola Cloud Gurus, y bienvenidos a esta conferencia.

Y esta conferencia trata sobre ProvisionedThroughputExceededException

y retroceso exponencial . Entonces empecemos.

Ahora, la provisión para la excepción de rendimiento excedido es una que podría ver en

su Tabla de DynamoDB si su tasa de solicitud es demasiado alta para el

de lectura y escritura aprovisionada en la tabla de DynamoDB .

Así por ejemplo,

si su aplicación realiza demasiadas solicitudes en DynamoDB

mesa ,

demasiadas solicitudes de lectura y escritura para que la tabla las maneje en función de su

de lectura y escritura. Ahora,

si está utilizando el SDK de AWS,

va a volver a intentar automáticamente las solicitudes hasta que tengan éxito.

Sin embargo, si no está utilizando el SDK,

vas a tener que configurar tu aplicación para hacer uno o ambos de los

siguiendo las cosas. En primer lugar, puede reducir la frecuencia de las solicitudes,

así que reduzca la cantidad de solicitudes simultáneas que se envían a DynamoDB

table , reducir la frecuencia de las solicitudes que se envían a la mesa,

y lo segundo que puede hacer es implementar un retroceso exponencial .

Ahora, ¿qué es el retroceso exponencial ? Bien,

muchos componentes en una red o en un sistema dado pueden generar errores

debido a que está sobrecargado con demasiadas solicitudes,

todos entrando al mismo tiempo.

Y ese podría ser cualquier componente que se te ocurra,

tal vez conmutadores de red, servidores DNS, balanceadores de carga,

etcétera .

Prácticamente cualquier parte de la red puede experimentar una sobrecarga debido a

demasiadas solicitudes simultáneas.

Y la forma habitual de lidiar con esto es implementar reintentos desde el cliente.

aplicación \_ Así por ejemplo,

para que el cliente siga intentando enviar una solicitud hasta que obtenga una respuesta.

Y todos los SDK de AWS, todos implementan reintentos automáticos,

que simplemente envía la solicitud nuevamente hasta que sea exitosa.

Pero además de estos simples reintentos,

todos los SDK de AWS también utilizan un retroceso exponencial .

Y con retroceso exponencial ,

esto significa que el solicitante utiliza esperas cada vez más largas entre

reintentos consecutivos para mejorar el control de flujo.

Por ejemplo, después de la primera solicitud fallida, podría esperar,

por ejemplo, 50 milisegundos antes de volver a intentarlo.

Y luego, si ese reintento falla,

entonces podría esperar 100 milisegundos hasta volver a intentarlo.

Y luego, si ese reintento falla,

podría esperar 200 milisegundos antes de volver a intentarlo.

Por lo tanto, continuará así hasta que la solicitud sea exitosa.

Y esto proporciona un control de flujo significativamente mejorado con la esperanza de que en algún momento

punto , el tráfico va a empezar a fluir libremente de nuevo,

y las solicitudes comenzarán a cumplirse nuevamente.

Pero si después de aproximadamente 1 minuto esto no funciona,

podría ser que el tamaño de su solicitud realmente exceda el

rendimiento para su capacidad de lectura y escritura en la tabla de DynamoDB .

Entonces, en este caso, puede valer la pena investigar su rendimiento aprovisionado.

Y si su carga de trabajo es principalmente Obtener solicitudes,

que pueda mejorar el rendimiento con DynamoDB Accelerator,

o DAX, o usando ElastiCache . Sin embargo,

si son principalmente las solicitudes de escritura las que están causando el problema,

entonces quizás eche un vistazo a aumentar su capacidad de escritura para la tabla de DynamoDB .

Entonces, ¿cuáles son nuestros consejos para el examen? Bien,

si ve el error ProvisionedThroughputExceeded ,

esto significa que la cantidad de solicitudes en su tabla de DynamoDB es demasiado alta.

El retroceso exponencial se puede utilizar para mejorar el control de flujo al volver a intentar el

solicitudes usando esperas progresivamente más largas. Sin embargo,

si su espera se acerca a 1 minuto,

vale la pena verificar la configuración de las unidades de capacidad de lectura y escritura y

viendo si necesitan ser ajustados. Y solo recuerda ese retroceso exponencial ,

no solo se aplica a DynamoDB .

En realidad, es una característica de todos los SDK de AWS,

y se aplica a muchos servicios dentro de AWS, por ejemplo,

Cubos de S3, si usa mucho CloudFormation , y SES, etcétera.

Y si no está utilizando el SDK de AWS,

entonces tendrá que manejar esto usted mismo en la configuración de su aplicación

o en el código de su aplicación.

Entonces, la forma de hacerlo es simplemente reducir la frecuencia de sus solicitudes y

implemente el retroceso exponencial usted mismo. Así que ese es el final de esta conferencia.

Si tiene alguna pregunta, hágamelo saber si no,

Siéntete libre de pasar a la siguiente lección. Gracias.