Lección 5
Almacenamiento de bases de datos en caché

¿Cuándo debería almacenar su base de datos en caché?

- Le preocupan los tiempos de respuesta para su cliente.
- Tiene un gran volumen de solicitudes que está inundando su base de datos.
- Desea reducir los costos de base de datos.

Como aprendió anteriormente en esta unidad, las consultas de bases de datos que consumen mucho tiempo y las consultas complejas pueden crear cuellos de botella en las aplicaciones.

Debe considerar almacenar su base de datos en caché en los siguientes casos:

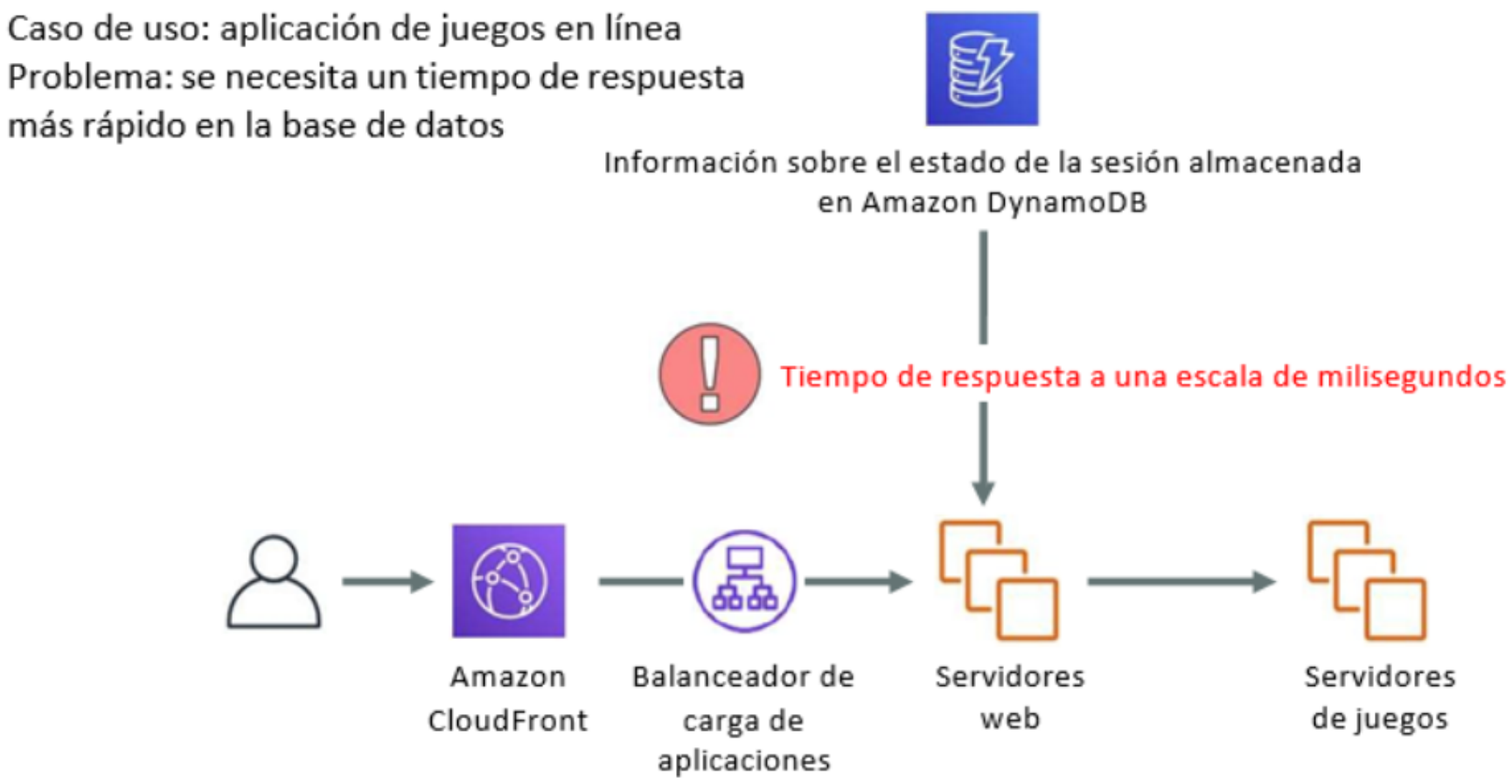
- Le preocupan los tiempos de respuesta para su cliente. Es posible que tenga cargas de trabajo en las que la latencia sea un factor importante que desee acelerar. El almacenamiento en caché puede ayudarlo a aumentar el rendimiento y reducir la latencia de la recuperación de datos, y, por lo tanto, mejorar el desempeño de la aplicación.

- Tiene un gran volumen de solicitudes que inundan su base de datos. Es posible que tenga una gran cantidad de tráfico y, por lo tanto, no obtenga el rendimiento que necesita para esa carga de trabajo. Colocar una capa de almacenamiento en caché junto a la base de datos puede aumentar el rendimiento y potenciar su desempeño.

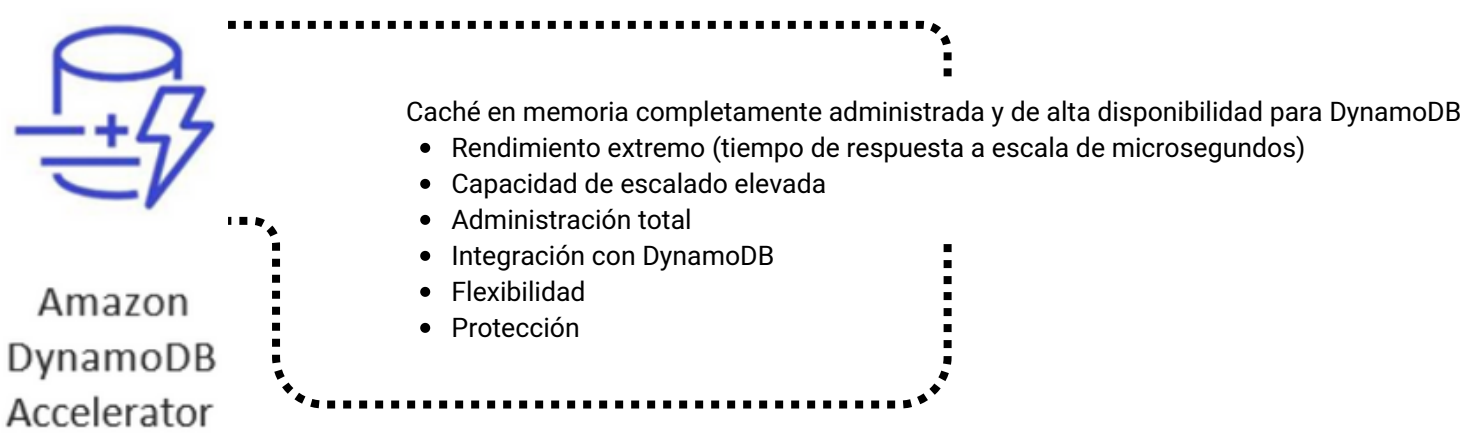
- Desea reducir los costos de base de datos. Tanto si los datos se distribuyen en una base de datos NoSQL basada en disco como si se escalan verticalmente en una base de datos relacional, el escalado para obtener una gran cantidad de lecturas puede ser costoso. Es posible que se necesite una serie de réplicas de lectura de bases de datos para que coincidan con lo que un único nodo de caché en memoria puede entregar en términos de solicitudes por segundo.

Una caché de base de datos complementa la base de datos principal eliminando la presión innecesaria que recae sobre ella, normalmente en forma de datos de lectura a los que se accede con frecuencia. La caché en sí puede estar en varias áreas, incluidas la base de datos o la aplicación, o puede aparecer como una capa independiente.

Uso de DynamoDB para información de estado



Considere una versión simplificada de la arquitectura para una aplicación de juegos en línea, en la que almacena información sobre el estado de la sesión en DynamoDB. En algunos casos, es posible que note que un tiempo de respuesta a escala de milisegundos no es lo suficientemente rápido para su aplicación. El almacenamiento en caché de la base de datos puede ayudar con este problema.



Amazon DynamoDB Accelerator (DAX) es una caché en memoria completamente administrada y de disponibilidad alta para DynamoDB.Ofrece un rendimiento hasta 10 veces superior, el cual se reduce de milisegundos a microsegundos, incluso con millones de solicitudes por segundo.

DAX se encarga de todo el trabajo pesado que se necesita para agregar aceleración en memoria a las tablas de DynamoDB. No es necesario administrar la invalidación de caché, el ingreso de datos ni la administración de clústeres.

DAX brinda los siguientes beneficios:

- Rendimiento extremo: DynamoDB ofrece una latencia uniforme de milisegundos de un solo dígito. Cuando DynamoDB y DAX se utilizan juntos, puede lograr tiempos de respuesta de tan solo microsegundos para millones de solicitudes por segundo correspondientes a cargas de trabajo de lectura intensiva.
- Capacidad de escalado elevada: DAX cuenta con escalado bajo demanda. Puede comenzar con un clúster DAX de tres nodos y agregar capacidad según sea necesaria, hasta alcanzar un clúster de 10 nodos.
- Administración total: al igual que DynamoDB, DAX está completamente administrada. DAX se encarga de tareas de administración, incluidos aprovisionamiento, instalación y configuración, implementación de parches de software y replicación de datos en nodos durante las operaciones de escalado. DAX automatiza las tareas administrativas comunes, como detección de errores, recuperación de errores e implementación de parches de software.
- Integración con DynamoDB: la API de DAX es compatible con DynamoDB y no es necesario realizar ningún cambio funcional en el código de la aplicación. Aprovisione un clúster de DAX y utilice el kit de desarrollo de software (SDK) de cliente DAX para dirigir las llamadas existentes a la API de DynamoDB al clúster de DAX. DAX se encarga del resto.

- Flexibilidad: puede aprovisionar un clúster de DAX para varias tablas de DynamoDB, varios clústeres de DAX para una sola tabla de DynamoDB o una combinación de ambos.

- Protección: DAX se integra completamente a los servicios de AWS para mejorar la seguridad. Puede utilizar AWS Identity and Access Management (IAM) para asignar credenciales de seguridad únicas a cada usuario y controlar el acceso de cada uno de ellos a los servicios y los recursos. Con Amazon CloudWatch, podrá ver la utilización de recursos, el rendimiento de las aplicaciones y el estado operativo de todo el sistema. La integración a AWS CloudTrail le permite registrar y auditar fácilmente los cambios en la configuración del clúster. DAX es compatible con Amazon Virtual Private Cloud (Amazon VPC), lo que ofrece un acceso seguro y sencillo desde sus aplicaciones existentes. El etiquetado brinda visibilidad adicional para ayudar a administrar los clústeres de DAX.

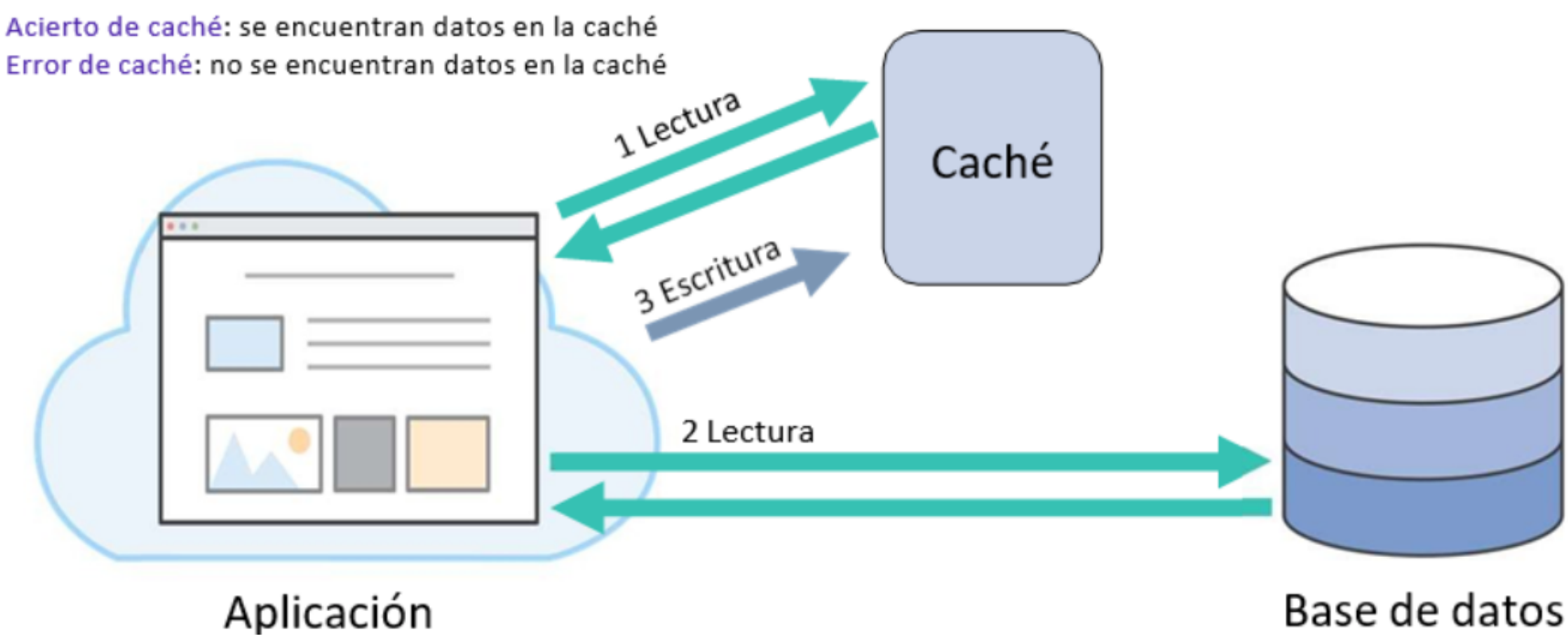
La recuperación de datos almacenados en caché reduce la carga de lectura en las tablas de DynamoDB existentes. Como resultado, esto puede reducir la capacidad de lectura aprovisionada y disminuir los costos operativos generales.

Para obtener más información acerca de DAX, consulte esta publicación del blog de base de datos de AWS.

Uso de DynamoDB con DAX para acelerar el tiempo de respuesta

En el ejemplo de la aplicación de juegos, la aceleración que se obtiene al agregar DAX a su arquitectura evita la necesidad de realizar algún cambio importante en el código del juego. De esta manera, se simplifica la implementación en su arquitectura. Lo único que debe hacer es volver a inicializar su cliente de DynamoDB con un nuevo punto de enlace que lleve a DAX. No se necesita hacer ningún cambio en el resto del código. DAX se encarga de la invalidación de caché y el ingreso de datos sin su intervención. Esta caché puede ayudar a acelerar la capacidad de respuesta cuando ejecute eventos que podrían provocar un pico en la cantidad de jugadores. Un ejemplo de esta clase de evento es una oferta de contenido descargable (DLC) estacional o una nueva versión en los parches.

Cachés remotas o laterales



DAX es una caché transparente. Otra forma de aplicar una implementación de caché de base de datos es usar una caché remota o lateral. Una caché lateral no está conectada directamente con la base de datos; en su lugar, se usa de manera adyacente a la base de datos. Las cachés laterales se suelen crear en almacenes de valores-clave NoSQL, como Redis o Memcached. Proporcionan cientos de miles de solicitudes, hasta un millón de solicitudes por segundo por nodo de caché.

Las cachés laterales se suelen utilizar para cargas de trabajo de lectura intensiva. Funcionan de la siguiente manera:

1. Para un determinado par de valor-clave, una aplicación primero intenta leer los datos de la caché. Si la caché contiene los datos (denominado acierto de caché), se devuelve el valor.

2. Si el par de valor-clave deseado no se encuentra en la caché (denominado error de caché), la aplicación obtiene los datos de la base de datos subyacente.

3. Es importante que los datos estén presentes cuando la aplicación los necesite de nuevo. Para asegurarse de que estén, el par de valor-clave que se obtiene de la base de datos se escribe en la caché.

Amazon ElastiCache

ElastiCache proporciona a las aplicaciones web un almacén de datos en memoria en la nube.

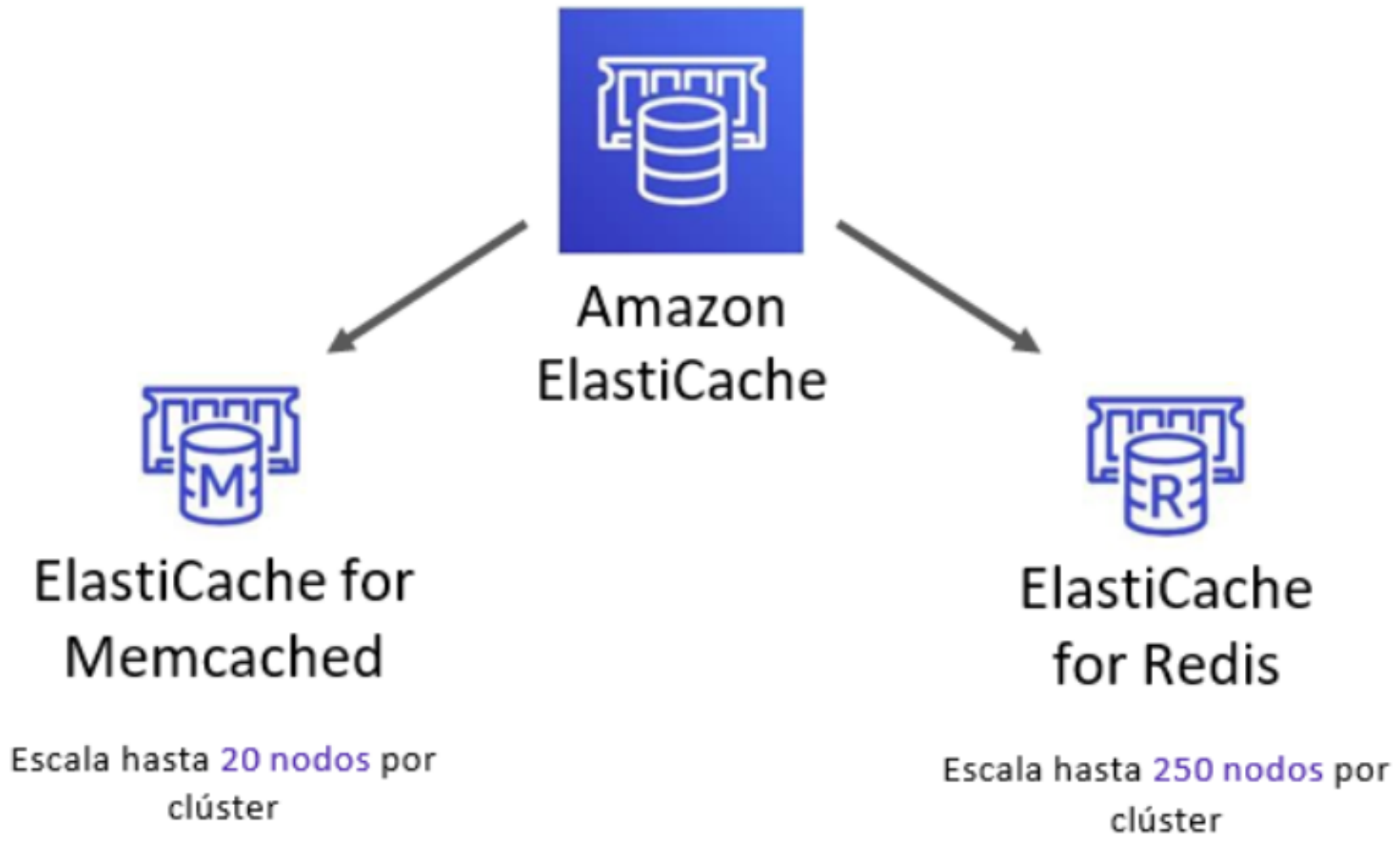
- Funciona como caché y almacén de datos en memoria.
- Ofrece un alto rendimiento.
- Está completamente administrado.
- Es escalable.
- Admite Redis y Memcached.

Amazon ElastiCache es una caché lateral que funciona como almacén de datos en memoria para admitir las aplicaciones más exigentes, las cuales requieren tiempos de respuesta inferiores a un milisegundo.

Con ElastiCache, ya no es necesario realizar tareas administrativas, como el aprovisionamiento de hardware, la implementación de parches de software, la instalación, la configuración, el monitoreo, la recuperación de errores y las copias de seguridad.

ElastiCache monitorea constantemente los clústeres para mantener las cargas de trabajo en funcionamiento, lo que le permite concentrarse en el desarrollo de aplicaciones de mayor valor. Amazon ElastiCache puede aumentar o reducir su escala horizontal, además de hacer crecer su escala vertical para satisfacer las demandas fluctuantes de las aplicaciones. El escalado de la memoria y la escritura se admite con particionamiento. Las réplicas se encargan de realizar el escalado de la lectura. ElastiCache admite dos bases de datos en memoria de código abierto: Redis y Memcached.

Redis y Memcached



ElastiCache for Memcached puede crecer hasta llegar a los 20 nodos por clúster. Por el contrario, ElastiCache for Redis puede llegar a los 250 nodos para aumentar el rendimiento del acceso a los datos. ElastiCache es compatible con Amazon VPC, lo que le permite aislar el clúster a los intervalos de direcciones IP que elija para los nodos.

ElastiCache se ejecuta en la misma infraestructura altamente confiable que utilizan otros servicios de AWS. ElastiCache for Redis brinda alta disponibilidad a través de implementaciones Multi-AZ con conmutación por error automática. Para las cargas de trabajo de Memcached, los datos se distribuyen entre todos los nodos del clúster. Por lo tanto, puede escalar horizontalmente para encargarse de más datos a medida que crezca la demanda.

Característica	Memcached	Redis
Latencia inferior a un milisegundo	Sí	Sí
Capacidad de escalar horizontalmente en escrituras y almacenamiento	Sí	No
Rendimiento de varios subprocesos	Sí	No
Estructuras de datos avanzadas	No	Sí
Organización y clasificación de conjuntos de datos	No	Sí
Mensajería de publicación/suscripción	No	Sí
Implementaciones Multi-AZ con conmutación por error automática	No	Sí
Persistencia	No	Sí

Los motores Memcached y Redis son cachés simples que se pueden utilizar para derivar la carga de la base de datos. Cada motor ofrece ciertas ventajas. Esta tabla compara algunas características clave de Memcached y Redis.

- **Latencia inferior a un milisegundo:** ambos motores ofrecen tiempos de respuesta inferiores a un milisegundo. Como almacenan los datos en la memoria, pueden leerlos con mayor rapidez que las bases de datos basadas en disco.

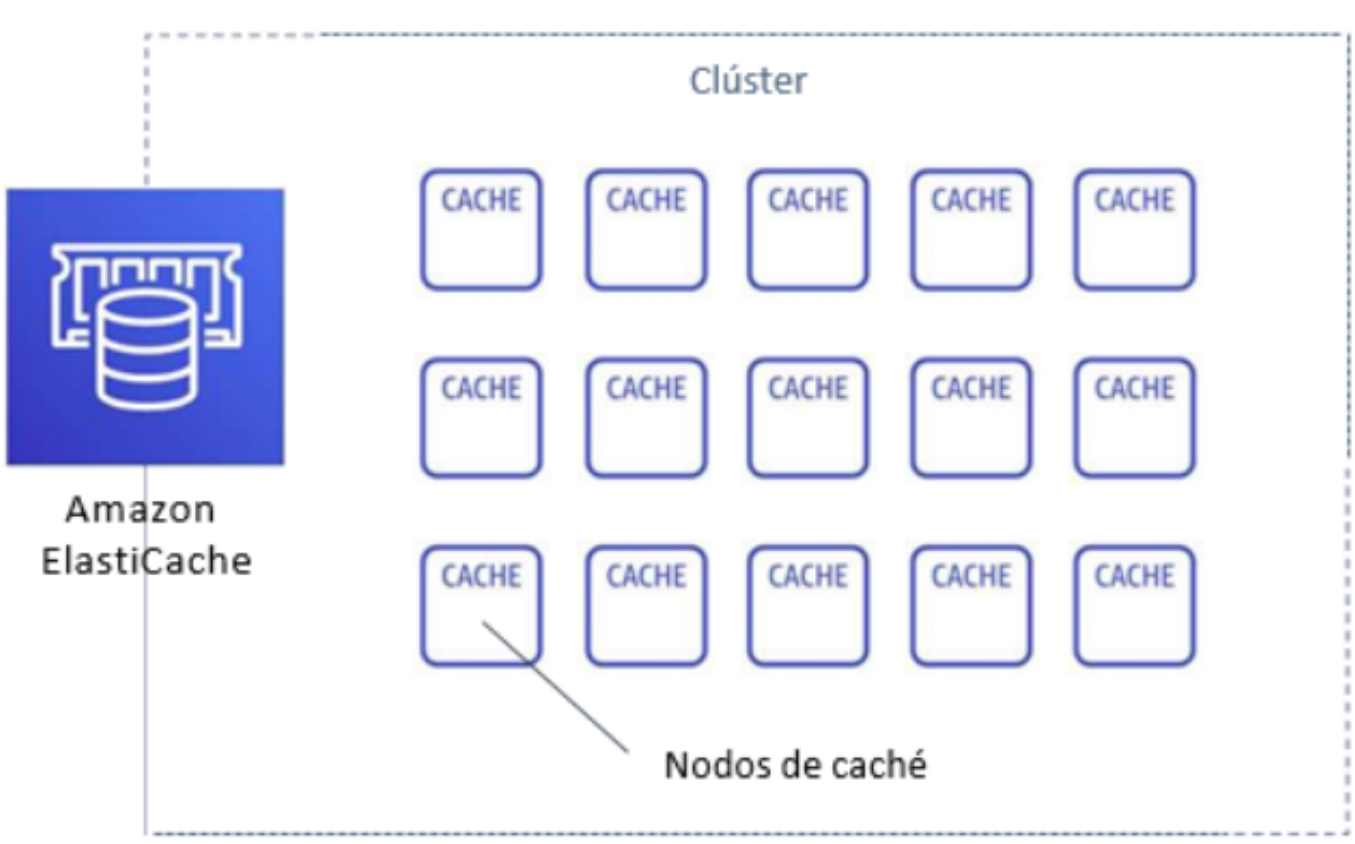
- **Capacidad de escalar horizontalmente:** Memcached le permite ampliar o reducir su escala horizontal, además de agregar y eliminar nodos a medida que la demanda del sistema aumente y disminuya.

- **Rendimiento de varios subprocesos:** como Memcached contiene subprocesos, puede utilizar varios núcleos de procesamiento, lo que significa que puede tolerar más operaciones mediante el aumento de la capacidad de cómputo.

- **Estructuras de datos avanzadas:** Redis admite tipos de datos complejos, como cadenas, hash, listas, conjuntos, conjuntos ordenados y mapas de bits.

- **Organización o clasificación de conjuntos de datos:** puede usar Redis para ordenar o clasificar conjuntos de datos en memoria. Por ejemplo, puede usar conjuntos ordenados de Redis para implementar una tabla de clasificación de un juego que mantenga una lista de jugadores ordenada por sus clasificaciones.
- **Mensajería de publicación/suscripción:** Redis admite mensajes de publicación o suscripción con coincidencia de patrones, que puede utilizar para salas de chat de alto rendimiento, flujos de comentarios en tiempo real, canales de redes sociales e intercomunicación entre servidores.
- **Implementaciones Multi-AZ con conmutación por error automática:** ElastiCache for Redis brinda alta disponibilidad a través de implementaciones Multi-AZ con conmutación por error automática, en caso de que se produzca un error en el nodo principal.
- **Persistencia:** Redis le permite conservar su almacén de claves. Por el contrario, el motor Memcached no admite la persistencia. Por ejemplo, probablemente haya un error en un nodo y se reemplace con otro nuevo nodo vacío, o bien, podría terminar un nodo o reducir el tamaño de uno. En este caso, se pierden los datos almacenados en la memoria caché.

Componentes de ElastiCache

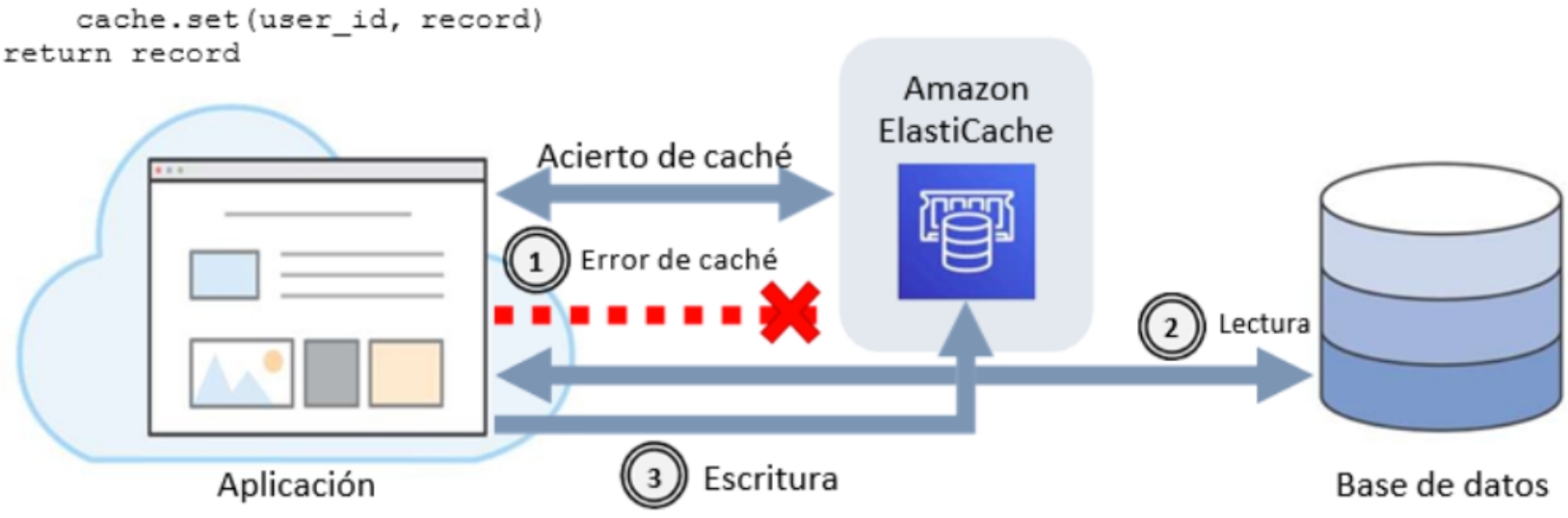


- Un nodo es el bloque más pequeño de una implementación de ElastiCache.
- Cada nodo tiene su propio nombre de DNS y puerto.
- Un clúster es una agrupación lógica de uno o más nodos.

Un nodo de caché es el componente básico más pequeño de una implementación de ElastiCache. Se trata de un segmento de tamaño fijo de RAM conectada a la red. Cada nodo ejecuta el motor que se eligió cuando se creó o modificó por última vez el clúster o el grupo de replicación. Cada nodo tiene su propio nombre de DNS y puerto. Puede existir aislado de otros nodos o agrupado con otros nodos, lo que también se conoce como clúster.

Estrategias de almacenamiento en caché: carga diferida

```
def get_user(user_id):
    # Check the cache
    record = cache.get(user_id)
    if record is None:
        # Run a DB query
        record = db.query("select * from users where id = ?", user_id)
        # Populate the cache
        cache.set(user_id, record)
    return record
```



Puede valerse de dos estrategias de almacenamiento en caché con ElastiCache.

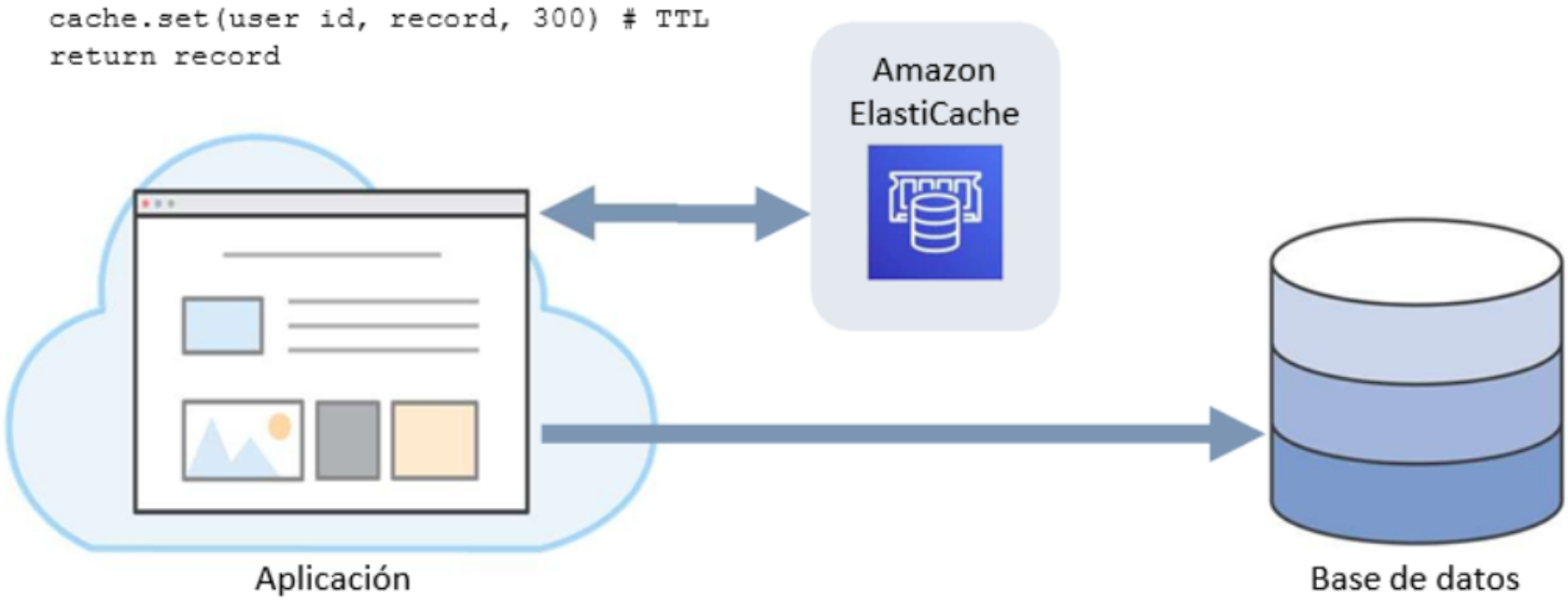
La primera estrategia, la carga diferida, es una estrategia de almacenamiento en caché que carga los datos en la caché solo cuando es necesario. En este caso, cuando su aplicación solicite datos, primero realizará una solicitud a la caché de ElastiCache. Si los datos existen en la caché y están actualizados, se produce un acierto de la caché y ElastiCache devuelve los datos a la aplicación. De lo contrario (en el caso de un error de caché), su aplicación solicita los datos del almacén de datos, que los devuelve a su aplicación. Luego, la aplicación escribe los datos en la caché para que puedan recuperarse más rápidamente la próxima vez que se los solicite.

Use la carga diferida cuando tenga datos que se leerán con frecuencia, pero se escribirán con poca frecuencia. Por ejemplo, en una aplicación web o móvil típica, el perfil de un usuario rara vez cambia, pero se accede a él en toda la aplicación. Es posible que una persona actualice su perfil solo unas cuantas veces al año. Sin embargo, tal vez se acceda al perfil decenas o cientos de veces al día, según el usuario.

La ventaja de la carga diferida es que solo se almacenan en caché los datos solicitados. Como la mayoría de los datos nunca se solicitan, la carga diferida evita llenar la caché con datos innecesarios. Sin embargo, un error de caché implica una penalización. Cada error de caché tiene como resultado tres viajes, lo que puede causar un retraso notable en la llegada de los datos a la aplicación. Además, si los datos se escriben en la caché solo cuando hay un error de caché, los datos en la caché pueden volverse obsoletos. La carga diferida no proporciona actualizaciones a la caché cuando se modifican los datos en la base de datos.

Estrategias de almacenamiento en caché: escritura inmediata

```
def save_user(user_id, values):
    # Save to DB
    record = db.query("update users…where id = ?", user_id, values)
    # Push into cache
    cache.set(user id, record, 300) # TTL
    return record
```

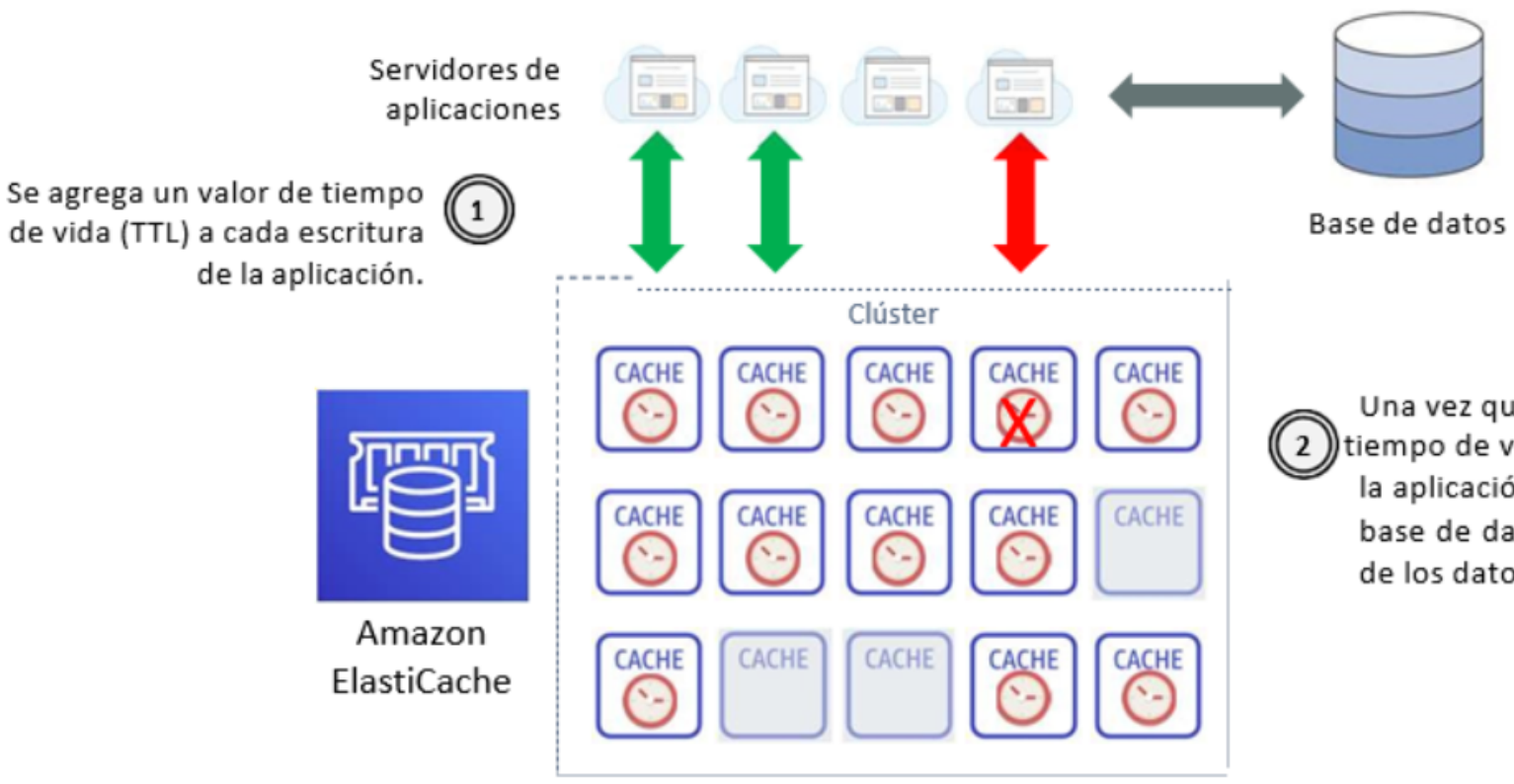


La estrategia de escritura inmediata es la segunda estrategia de almacenamiento en caché. Agrega datos o los actualiza en la caché cuando se escriben en la base de datos. Utilice una estrategia de almacenamiento en caché de escritura inmediata cuando tenga datos que deban actualizarse en tiempo real. Este enfoque es proactivo; puede evitar errores de caché innecesarios cuando tenga datos a los que está seguro que se accederá. Un buen ejemplo de esta situación es cualquier tipo de agrupación, como una clasificación de los 100 mejores juegos, las 10 noticias más populares o recomendaciones. Como estos datos normalmente se actualizan a través de un fragmento específico de código de la aplicación o del trabajo en segundo plano, es sencillo actualizar la caché también.

La ventaja de este enfoque es que aumenta la probabilidad de que su aplicación encuentre ese valor en la caché cuando lo busque. La desventaja es que es probable que se almacenen en caché datos que no necesita, por lo que se puede incurrir en costos adicionales.

En la práctica, ambos enfoques se usan juntos, así que es importante que comprenda la frecuencia con la que se modifican los datos y utilice los TTL adecuados.

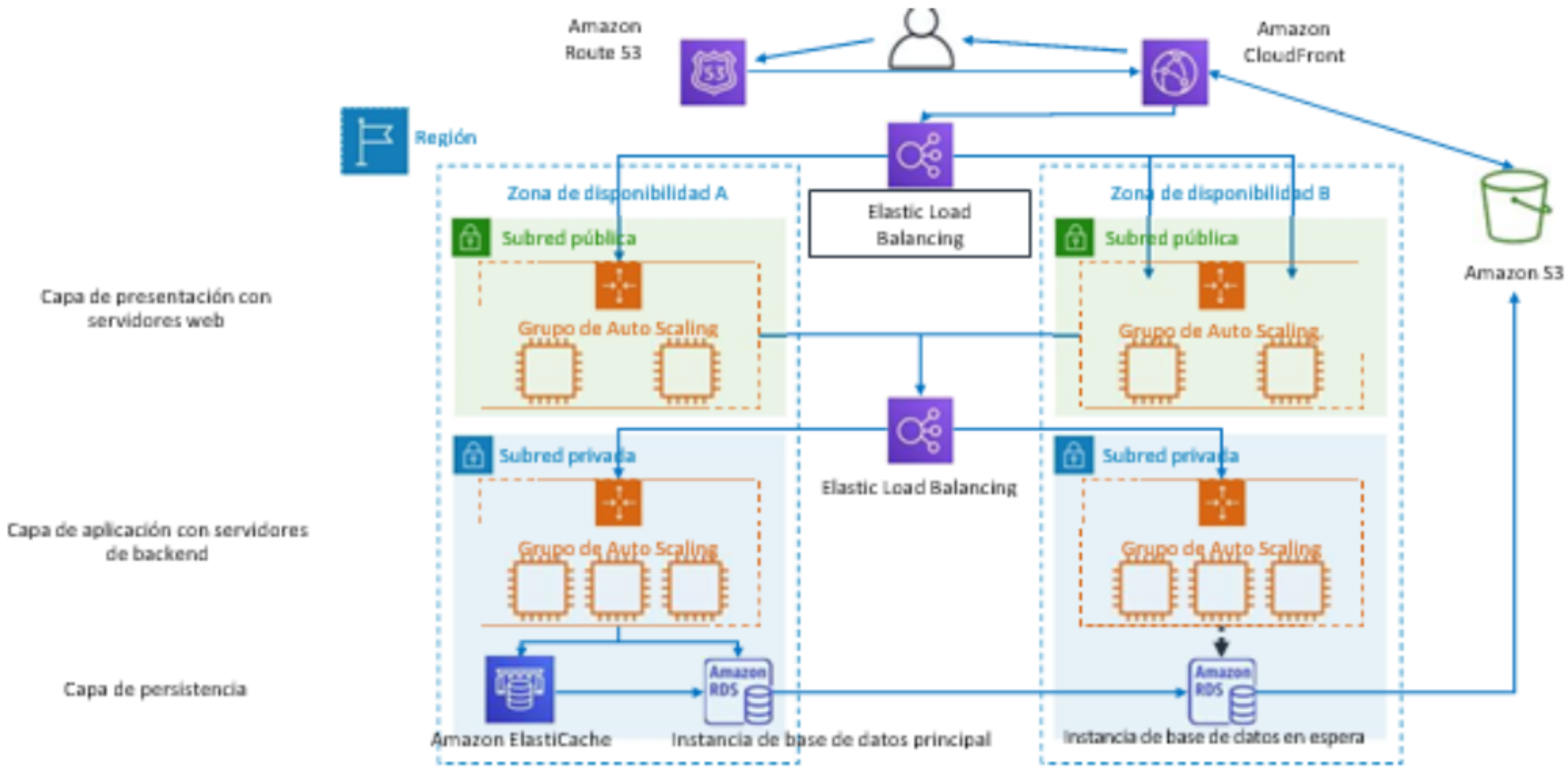
Incorporación de TTL



La carga diferida permite que existan datos obsoletos. La escritura inmediata garantiza que los datos estén siempre actualizados, pero este enfoque podría llenar la caché con datos innecesarios.

Al agregar un valor de TTL a cada escritura, disfruta las ventajas de cada estrategia y evita saturar la caché con datos. TTL es un valor o una clave enteros que especifica el número de segundos o milisegundos, en función del motor en memoria, que deben transcurrir para que la clave venza. Cuando una aplicación intenta leer una clave vencida, se trata como si los datos no se encontraran en la caché. Como resultado, se consulta la base de datos y se actualiza la caché. De esta manera, los datos no se vuelven demasiado obsoletos y los valores de la caché se actualizan de manera ocasional a partir de la base de datos.

Arquitectura de alojamiento web de tres niveles



Un caso en el que es posible que desee utilizar Amazon ElastiCache es en una arquitectura tradicional de alojamiento web de tres niveles. En este caso, debería ejecutar una aplicación web pública a la vez que mantiene los servidores de backend privados en una subred privada. Puede crear una subred pública para los servidores web que tengan acceso a Internet. Al mismo tiempo, puede ubicar su infraestructura de backend en una subred privada sin acceso a Internet. El nivel de base de datos de su infraestructura de backend podría incluir instancias de base de datos de Amazon Relational Database Service (Amazon RDS) y un clúster de ElastiCache que proporciona la capa en memoria.

En este diagrama de arquitectura de alojamiento web:

- Amazon Route 53 le permite asignar el nombre de DNS de su ápex de zona (como
- ejemplo.com) al nombre de DNS del balanceador de carga.
- Amazon CloudFront ofrece almacenamiento en caché perimetral para grandes volúmenes de contenido.
- Un balanceador de carga distribuye el tráfico entre los servidores web en grupos de Auto Scaling de la capa de presentación.
- Otro balanceador de carga distribuye el tráfico entre los servidores de aplicaciones de backend en los grupos de Auto Scaling que se encuentran en la capa de aplicación.
- Amazon ElastiCache proporciona una caché de datos en memoria para la aplicación, que elimina la carga del nivel de base de datos.

Estos son algunos de los aprendizajes clave de esta lección

- Una caché de base de datos complementa la base de datos principal, ya que elimina presión innecesaria que recae sobre ella, normalmente en forma de datos de lectura a los que se accede con frecuencia.
- DAX es una caché en memoria completamente administrada y de alta disponibilidad para DynamoDB que ofrece un rendimiento hasta 10 veces superior, el cual se reduce de milisegundos a microsegundos.
- Amazon ElastiCache es una caché lateral que funciona como almacén de datos en memoria para admitir las aplicaciones más exigentes, las cuales requieren tiempos de respuesta inferiores a un milisegundo.

Ha llegado el momento de hacer un repaso de la unidad y concluir con una evaluación de conocimientos

A modo de resumen, en esta unidad, aprendió a hacer lo siguiente:

- reconocer cómo el almacenamiento de contenido en caché puede mejorar el rendimiento de las aplicaciones y reducir la latencia
- crear arquitecturas que utilicen Amazon CloudFront para almacenar el contenido en caché
- saber cómo diseñar arquitecturas que utilicen ubicaciones perimetrales para la distribución y la protección contra ataques de denegación de servicio distribuidos (DDoS)
- reconocer cómo se relaciona la administración de sesiones con el almacenamiento en caché
- describir cómo diseñar arquitecturas que utilicen Amazon ElastiCache