Disponibilidad

Resolución conceptual, takeaways y buenas prácticas

¡Felicitaciones! A esta altura, seguramente ya hayan logrado completar el primer desafío.

Como habrás notado, este tipo de desafíos permiten varias soluciones posibles, algunas de las cuales pueden ser más adecuadas que otras, dependiendo de las restricciones específicas que enfrentamos. A lo largo de este curso, descubriremos que mejorar un aspecto de la Ingeniería de Confiabilidad del Sitio (SRE) a veces puede tener un impacto adverso en otro aspecto relacionado. A continuación analizaremos la estrategia de resolución que hemos identificado como la más eficiente. Y, por supuesto, si crees que tu solución es más efectiva que la nuestra, ¡estamos ansiosos por conocerla y aprender de tu enfoque!

✅ Resolución conceptual

Ante un llamado a una App que devuelve error el 50% de las veces, se necesita que esta tasa de error sea de a lo sumo un 5%. Para resolver este desafío se necesita conocer e implementar el patrón **Retry** para volver a llamar a la App en caso de que haya un error antes de dar una respuesta al usuario. Entonces, la probabilidad de falla compuesta de dos llamados consecutivos es del 25% (0.5 x 0.5 = 0.25). De esta manera se puede llegar a la conclusión de que si se quiere tener un error menor al 5% uno puede repetir la llamada hasta satisfacer dicha condición. En particular, cuando se repite el llamado 5 veces se encuentra que la app fallará en el 3.125% de la veces y, por tanto, se cumplirá con el objetivo planteado. Es vital, luego de hacer este cálculo estimado, probar empíricamente que la hipótesis de solución se valida en los hechos.

Patrón Retry

Para lograr la disponibilidad deseada, el patrón Retry suele ser de gran importancia. El mismo permite al sistema volver a intentar un llamado en caso que este falle de manera transitoria, como interrupciones breves de red o sobrecargas temporales de un servicio. En lugar de considerar inmediatamente una operación como fallida, el sistema reintenta

automáticamente la misma operación después de un breve período. Un aspecto crucial en SRE es encontrar el equilibrio correcto en la política de reintentos.

Demasiados reintentos rápidos pueden sobrecargar aún más un sistema ya comprometido, mientras que pocos reintentos o intervalos de tiempo muy largos pueden degradar la experiencia del usuario, impactando negativamente en la performance del sitio. A su vez es fundamental monitorizar cuidadosamente los patrones de reintento para identificar y resolver problemas subyacentes en la infraestructura o en la aplicación. Una alta tasa de reintentos puede ser un indicador de problemas más profundos que necesitan ser abordados de otras maneras.

En resumen, el patrón Retry es una herramienta esencial para mantener la operatividad y la calidad del servicio en entornos inciertos y propensos a errores. Se utiliza con cuidado y en conjunto con otras estrategias, como el **backoff**, para asegurar que los sistemas sean tanto resilientes como eficientes.

Es importante tener en cuenta que existen circunstancias en las que la función Retry no es útil, como por ejemplo:

- Si un error es definitivo o no transitorio, como un error de validación de datos o una falla crítica de sistema, el reintento no es útil. En estos casos es mejor manejar el error sin reintentar, ya que el problema no se resolverá con intentos adicionales.
- Donde los reintentos pueden ser costosos en términos de recursos o tiempo (por ejemplo, operaciones que consumen mucha CPU o ancho de banda), puede ser preferible abordar el error de manera diferente, como alertar a un operador humano o usar una estrategia de fallback.

Función Backoff

La función backoff solo tiene sentido dentro de un Retry y permite aumentar el tiempo entre cada llamado recurrente del Retry. Es especialmente útil cuando hay riesgo de que el sistema se sobrecargue en una red congestionada, ya sea por ser un servicio muy demandado y/o que supone una alta exigencia a los servidores. Hay situaciones en las que aplicar la función backoff no es necesario o incluso puede ser contraproducente:

- En sistemas donde la entrega está garantizada o gestionada por otro mecanismo (como las colas de mensajes con confirmación de entrega), el backoff puede no ser necesario, ya que el sistema subyacente maneja los reintentos y la fiabilidad.
- En aplicaciones de tiempo real donde la respuesta inmediata es crítica (como sistemas de control industrial o aplicaciones de trading en tiempo real), el backoff

puede no ser adecuado, ya que incluso un pequeño retraso puede ser inaceptable.

- Limitaciones de Acuerdos de Nivel de Servicio (SLAs): Si los SLAs establecen una respuesta o resolución dentro de un marco de tiempo específico, el backoff puede no ser viable, ya que aumenta el tiempo total de respuesta.
- Cargas de Trabajo Predecibles y Estables: En ambientes donde la carga de trabajo y el tráfico son predecibles y estables, y el sistema está dimensionado adecuadamente para manejar esta carga, la necesidad de una lógica de backoff es mínima.
- Contexto de Uso Único o Transacciones Críticas: En operaciones de uso único o transacciones críticas, donde un fallo debe ser inmediatamente notificado o manejado de manera especial, el backoff puede no ser apropiado.
- Pruebas de Estrés o Simulación de Cargas: Durante las pruebas de estrés o simulaciones para evaluar la capacidad de un sistema, los reintentos inmediatos pueden ser útiles para generar una carga intensa y evaluar cómo responde el sistema.

En general, la decisión de aplicar o no la función de backoff depende del contexto específico, la naturaleza del error, y las características del sistema y de la aplicación. Es importante evaluar si el backoff añadirá valor y mejorará la resiliencia del sistema en un escenario dado.

✅ Buenas prácticas

Las buenas prácticas se refieren a métodos y enfoques recomendados para abordar desafíos de manera eficiente y efectiva. En este caso, podemos resumir las buenas prácticas incorporadas a través de la resolución del primer desafío en los siguientes puntos:

Adaptabilidad de Soluciones a Contextos Específicos: Cada desafío en el ámbito de SRE es único y, por lo tanto, requiere un análisis y una solución a medida. Es crucial evitar la aplicación de soluciones estándar sin considerar las particularidades del problema en cuestión. Siempre reflexione sobre la necesidad y el propósito de cada línea de código que escriba. Por ejemplo, en nuestro primer desafío, descubrimos que no era necesario emplear la función backoff. Este enfoque selectivo y reflexivo garantiza que cada solución sea tan eficiente y efectiva como sea posible, adaptada específicamente al problema que se está abordando.

Combinación de Análisis Teórico y Validación Experimental: Adopte un enfoque similar al del método científico para resolver problemas en SRE, buscando un equilibrio entre los cálculos teóricos y la validación práctica. Esto significa formular hipótesis basadas en

una comprensión teórica y luego probar estas hipótesis mediante experimentación en el entorno real. La retroalimentación obtenida de estas pruebas prácticas es invaluable para refinar su enfoque y lograr soluciones más robustas y confiables. Este proceso iterativo de teorización y experimentación asegura una comprensión profunda del problema y una solución más efectiva.

🎯 Takeaways

Los takeaways son conclusiones o datos específicos obtenidos de la experiencia que pueden informar decisiones futuras o establecer estándares. Para este desafío es bueno llevarnos que la tasa de disponibilidad que busca alcanzar mercado libre es de 99.95%. Este porcentaje de uptime significa que la plataforma está operativa y disponible para los usuarios durante el 99,95% del tiempo. Traducido a términos más concretos, implica que en un año (que tiene 8.760 horas), el tiempo de inactividad máximo total sería de aproximadamente 4,38 horas.

Este indicador sirve como un benchmark útil al considerar objetivos de disponibilidad en proyectos de SRE. Alcanzar y mantener un nivel alto de uptime como el de Mercado Libre requiere prácticas rigurosas de monitoreo, automatización, respuesta rápida a incidentes, y una arquitectura robusta y escalable. Este dato también refleja la importancia de equilibrar la innovación y el lanzamiento de nuevas características con la estabilidad y confiabilidad del sistema.

Muchas compañías consideran un uptime de 99.9% a 99.99% como adecuado y suficiente para sus necesidades, equilibrando el costo y la complejidad de alcanzar niveles más altos de disponibilidad con las necesidades de su negocio y sus usuarios. El máximo estándar en la industria es conocido como "Five Nines", que se refiere a un uptime de 99.999%. Este nivel de disponibilidad implica que el sistema solo está inactivo o no disponible durante aproximadamente 5.26 minutos al año. Alcanzar y mantener este estándar es extremadamente desafiante y requiere una infraestructura altamente confiable y sofisticadas prácticas de ingeniería de confiabilidad del sitio (SRE).

✍️ En tu próximo encuentro en vivo

En el próximo encuentro, trabajarás sobre el concepto de Eficiencia y Performance. Será con la misma metodología que has aplicado hasta ahora.

Recuerda que tu grupo de trabajo cambiará y tendrás la oportunidad de cooperar con otros integrantes que sumarán nuevas experiencias y conocimientos.