Retentativas

Repita operações transitórias com orçamento limitado

Pré-requisitos

Conheça falhas de transporte, idempotência e filas. Use Node.js 24. O laboratório calcula o início das chamadas de um consumidor que envia uma notificação. Os tempos são simulados e não envolvem espera real.

Classifique a falha pelo contrato

Um timeout informa que a resposta não chegou no prazo. Antes de repetir, confirme se há chave de idempotência ou consulta de resultado. Uma indisponibilidade temporária pode justificar outra tentativa. Dados rejeitados por regra permanente exigem correção. Se o provedor indicar quando tentar novamente, respeite essa indicação dentro do orçamento.

Combine três limites [1]

Use timeout por chamada, prazo total e número máximo de tentativas. Cada um limita uma parte do trabalho. Uma chamada com tempo disponível insuficiente pode exigir um timeout menor ou ser adiada. Neste exercício, o consumidor só inicia se ainda houver espaço para o timeout inteiro.

Espalhe as chamadas com jitter [1]

Backoff aumenta a espera depois de cada falha até um teto. No full jitter, sorteie uma espera entre zero e o teto daquela etapa. Isso reduz a chance de vários consumidores repetirem juntos quando a dependência está saturada. Use dados fixos de sorteio no teste para obter resultados reproduzíveis.

Escolha uma camada responsável

Três tentativas em cada uma de duas camadas podem gerar nove chamadas à dependência. Coordene a política do SDK, do consumidor e da fila. Inclua esperas no cálculo de visibilidade e deixe recuperação de longo prazo a cargo da fila ou de um agendamento durável.

Retentativas

Exercício resolvido • Calcule os próximos inícios

Enunciado e dados de partida

Use no máximo três chamadas, timeout de 400 ms e orçamento de 1.500 ms. As duas primeiras falham após 400 ms; a terceira, se iniciada, responde em 200 ms. A espera tem tetos de 200 e 400 ms. Compare sorteios fixos de 0,5 e 0,9. Para a falha permanente, use 25 ms.

1. Salve retentativas.mjs

```
import assert from 'node:assert/strict';

function planejar(respostas, sortear) {
  const inicio = [];
  let agora = 0;
  const limite = 1500;
  const timeout = 400;
  for (const [i, resposta] of respostas.slice(0, 3).entries()) {
    if (agora + timeout > limite) break;
    inicio.push(agora);
    agora += resposta.ms;
    if (resposta.ok || !resposta.transitoria) break;
    if (i < 2) {
      const teto = Math.min(200 * 2 ** i, 800);
      agora += Math.floor(sortear() * teto);
    }
  }
  return inicio;
}

const erro = { ms: 400, ok: false, transitoria: true };
const sucesso = { ms: 200, ok: true };
const invalido = { ms: 25, ok: false, transitoria: false };
assert.deepEqual(planejar([erro, erro, sucesso], () => 0.5), [0, 500, 1100]);
assert.deepEqual(planejar([erro, erro, sucesso], () => 0.9), [0, 580]);
assert.deepEqual(planejar([invalido, sucesso], () => 0.5), [0]);
console.log('Espera, prazo e falha definitiva conferidos');
```

2. Execute node retentativas.mjs

O programa imprime ‘Espera, prazo e falha definitiva conferidos’. O sorteio injetado representa um valor em [0, 1). O simulador recebe durações que já respeitam o timeout.

Retentativas

Confira o orçamento e o motivo de parada

3. Resolva o cenário de 0,5

A tentativa 1 inicia em 0 e termina em 400. A espera é 200 × 0,5 = 100, então a segunda começa em 500 e termina em 900. A espera seguinte é 400 × 0,5 = 200. A terceira começa em 1.100, tem orçamento completo até 1.500 e termina em 1.300.

4. Resolva o cenário de 0,9

A primeira espera é 180 e a segunda chamada começa em 580. Ela termina em 980. A segunda espera é 360 e a chamada seguinte começaria em 1.340. O prazo comporta só 160 ms, portanto o simulador para. A falha permanente para após uma chamada, com orçamento ainda disponível.

Caso de erro • Confira apenas o instante de início

Troque agora + timeout > limite por agora >= limite. A terceira chamada do segundo cenário começa em 1.340 e termina em 1.540, acima do prazo. A asserção de inícios falha. Restaure o cálculo com reserva do timeout completo.

Critérios para conferir

Compare a lista de instantes, a decisão de não repetir falhas permanentes e o orçamento final. O teste só planeja chamadas e não demonstra cancelamento real. Na implementação, controle o prazo com relógio monotônico quando disponível e propague o tempo restante para a dependência.

Referências primárias

[1] AWS Builders' Library • Timeouts, retentativas, backoff e jitter

Retentativas

Limite as retentativas para evitar sobrecarga

Escolha um único ponto de retentativa

Se cada camada repetir até três tentativas, uma chamada de nível superior pode multiplicar as chamadas às dependências. Limite as tentativas, aplique atraso crescente com aleatoriedade e respeite um prazo total.

Exercício adicional

Três camadas fazem até três tentativas, incluindo a inicial. Qual é o máximo no serviço final?

Resolução

Em uma sequência em que todas esgotam as tentativas, são $3 \times 3 \times 3 = 27$. Concentre a retentativa na camada que consegue classificar a falha e repetir a operação com segurança.

Ao assistir ao vídeo

Compare o intervalo fixo com esperas exponenciais aleatórias.

Vídeo complementar

Retries, Backoff, and Jitter (em inglês)