Avaliação Quantitativa do Padrão Circuit Breaker em Microsserviços

Um Estudo Empírico sobre Resiliência e Performance

Humberto Laff

Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE)

Dezembro de 2024

Agenda

Contextualização

Era dos Microsserviços

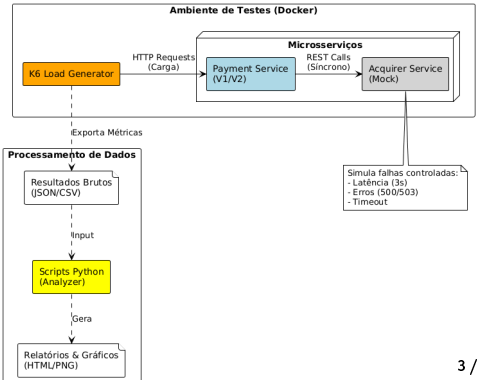
- Arquitetura predominante em sistemas de grande escala
- Comunicação síncrona (REST/HTTP) como padrão
- Flexibilidade + Escalabilidade + Deploy independente

O Preço da Comunicação Síncrona

- **Acoplamento temporal** entre serviços
- Requisições bloqueadas aguardando resposta
- Timeout = recursos desperdiçados

Custo do Downtime

Em sistemas bancários:
US$ 5.600 - US$ 9.000/min



O Problema: Falhas em Cascata

O que acontece quando uma dependência falha?

Cenário de Falha

```
[Usuário]
    ↓
[servico-pagamento]
    ↓ BLOQUEADO
[servico-adquirente]
    ↓ TIMEOUT
[Gateway Externo]
```

Consequências

1. **Thread Pool Starvation**
   Threads bloqueadas aguardando
2. **Efeito Dominó**
   A falha de C derruba B e A
3. **Sistema Inutilizável**
   Cascata de falhas

Problema: Falhas em cascata causadas por comunicação síncrona entre microsserviços

O Padrão Circuit Breaker

Analogia: Funciona como um **disjuntor elétrico**

- Detecta condições anormais
- Interrompe o fluxo para proteger o sistema
- Permite recuperação automática

Máquina de Estados:

1. **FECHADO**: Operação normal
2. **ABERTO**: Fail-fast + Fallback
3. **SEMIABERTO**: Período de teste

Configuração Utilizada

- `failureRateThreshold: 50%`
- `slidingWindowSize: 10 req`
- `waitDurationInOpenState: 10s`
- `permittedCallsInHalfOpen: 3`

Degradação Graciosa

HTTP 500 $\rightarrow$ HTTP 202
"Pagamento agendado"

Metodologia: Abordagem Experimental

Stack Tecnológico

- Java 17, Spring Boot 3
- Resilience4j 2.1.0
- Docker Compose
- Grafana k6 (load testing)

Versões Comparadas

V1 Baseline (sem resiliência)

V2 Circuit Breaker + Fallback

V3 Retry com Backoff

5 Cenários de Teste

Normal	100% saudável
Catástrofe	100% falha por 5min
Degradação	5% $\rightarrow$ 50% falhas
Rajadas	3 ondas de falha
Indisp.	75% offline

Volume de Dados

+380.000 requisições
analisadas estatisticamente

Resultados Consolidados

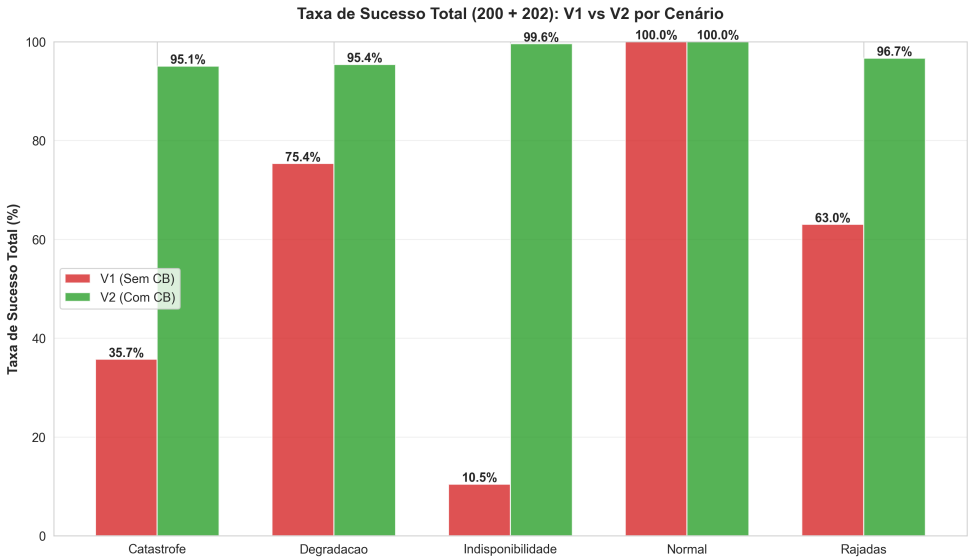
Comparação de Disponibilidade Percebida: V1 vs V2

Cenário	V1 (Baseline)	V2 (CB)	Ganho
Catástrofe	35,7%	**95,1%**	+59,3pp
Degradação	75,4%	**95,4%**	+20,0pp
cinufpe!15 Indisponibilidade	10,5%	**99,6%**	**+89,1pp**
Rajadas	63,0%	**96,7%**	+33,6pp
Normal	100,0%	100,0%	0pp

Resultado Principal

No cenário de **Indisponibilidade Extrema**, o Circuit Breaker transformou um sistema com **10,5%** de disponibilidade em um com **99,6%** — uma **melhoria de 9,5x**!

Visualização dos Resultados



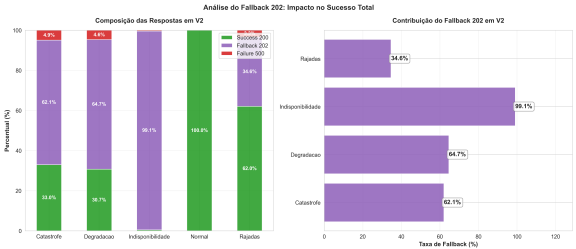
Impacto do Fallback

Transformação de Erros

- V1: HTTP 500 (erro fatal)
- V2: HTTP 202 (fallback)

Contribuição do Fallback:

- Catástrofe: 62,1%
- Rajadas: 34,6%
- Indisponibilidade: **99,1%**

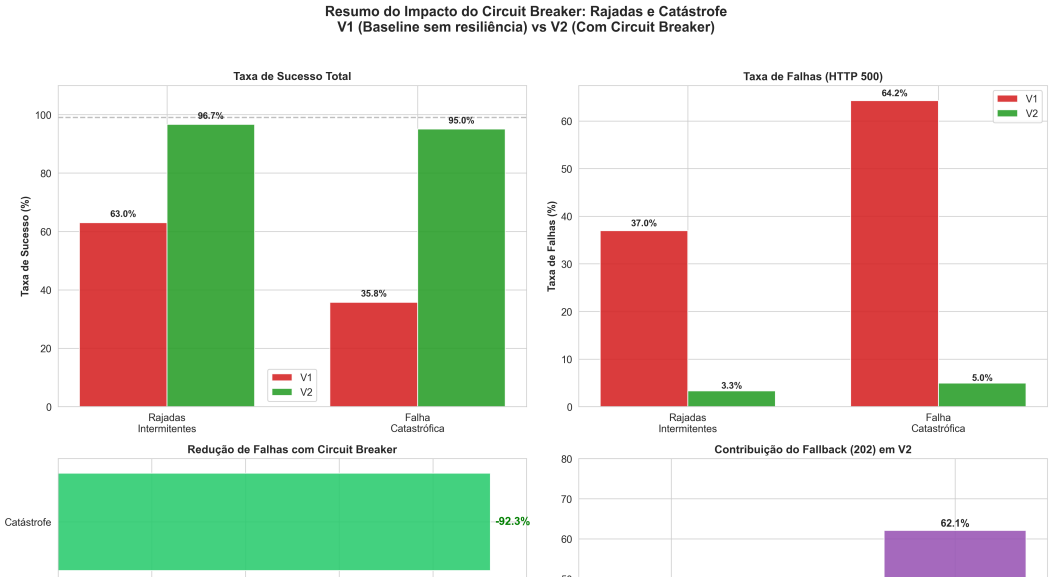


Redução de Downtime

Catástrofe: 8,37min → 0,64min

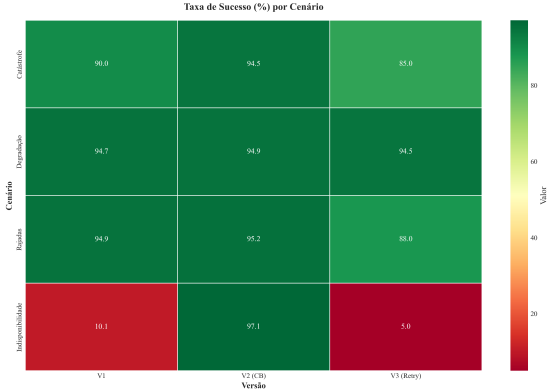
Redução de 92%

Análise Consolidada: Rajadas e Catástrofe



Validação Estatística

Métrica	Valor
Teste t (p-value)	$p < 0{,}0001$
cinufpe!15 **Cohen's d**	**1,078**
ANOVA (F)	546,79
Eta-quadrado (η^2)	0,267
IC 95% (V1)	[495; 508] ms
IC 95% (V2)	[400; 410] ms



Interpretação

Cohen's d = 1,078
= Efeito **GRANDE**
(limiar: d > 0,8)

Por que Retry (V3) não é Suficiente?

Cenário	V1	V3	V2
Indisp.	10,5%	15,7%	**99,6%**
Catástrofe	35,7%	52,8%	**95,1%**
Rajadas	63,0%	77,7%	**96,7%**

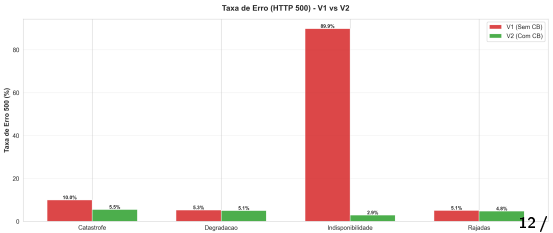
Problemas do Retry

1. Não melhora em falhas persistentes
2. Aumenta latência
3. Pode amplificar carga (3x)

Recomendação

Retry deve ser usado **dentro** do Circuit Breaker, não como substituto.

CB fecha → fallback imediato
CB aberto → retry interno



Contribuições do Trabalho

1. **Evidência Empírica Robusta**
   Dados quantitativos de 5 cenários realistas (+380.000 req)
2. **Comparação entre Padrões**
   CB (V2) vs Retry isolado (V3) — demonstrou superioridade do CB
3. **Análise Estatística Rigorosa**
   Cohen's d = 1,078 (efeito grande), ANOVA significativa
4. **Metodologia Reprodutível**
   Framework Docker + k6 disponível para replicação
5. **Quantificação de Benefícios**
   Até +**89pp** na disponibilidade, -**99,5%** nas falhas

Limitações e Trabalhos Futuros

Limitações

- POC simplificada
- Ambiente local
- Configuração fixa do CB
- Carga sintética (k6)

Trabalhos Futuros

- Combinação CB + Retry
- Ambiente cloud distribuído
- Múltiplas dependências
- CB adaptativos com ML
- Service Meshes (Istio)

Conclusão

"O Circuit Breaker transformou um sistema inutilizável (10,5%) em um altamente disponível (99,6%).
Isso representa uma melhoria de 9,5x."

CB é essencial
para microsserviços síncronos críticos

Retry sozinho
não é suficiente para falhas persistentes

Fallback
preserva a experiência do usuário

Obrigado!

Perguntas?

Humberto Laff
hlaff@cin.ufpe.br
Centro de Informática (CIn) - UFPE