Avaliação Experimental de Tolerância a Falhas e Resiliência em Microsserviços na Nuvem Privada com Engenharia do Caos, NetEm e Scaphandre

Ivan J. S. Pereira[1], **Gustavo Callou**[1]

[1] Universidade Federal Rural de Pernambuco (UFRPE)
Recife – PE – Brasil

ivan.saboia@ufrpe.br, gustavo.callou@ufrpe.br

Abstract. *The microservices architecture has become predominant due to its scalability and agility; however, it introduces significant challenges in fault management within private cloud environments. This work presents an experimental methodology to evaluate fault tolerance and energy consumption of the TeaStore application, deployed in Docker containers on virtual machines managed by OpenNebula. The approach combines Chaos Engineering techniques, including network fault injection (via NetEm) and container faults (via Pumba), under workloads generated by Apache JMeter, along with energy measurement using Scaphandre in a real physical testbed. Three scenarios were evaluated: baseline, network degradation, and critical service failure. The results show that faults can increase the energy cost per request by up to 5.6x, underscoring the importance of resilience not only for availability but also for sustainability.*

Resumo. *A arquitetura de microsserviços tornou-se predominante devido à sua escalabilidade e agilidade, mas introduz desafios significativos na gestão de falhas em ambientes de nuvem privada. Este trabalho apresenta uma metodologia experimental para avaliar a tolerância a falhas e o consumo energético da aplicação TeaStore, implantada em contêineres Docker sobre máquinas virtuais gerenciadas pelo OpenNebula. A abordagem combina Engenharia do Caos, com injeção de falhas de rede (via NetEm) e de contêineres (via Pumba), sob carga gerada pelo Apache JMeter, além da medição de energia com Scaphandre em um testbed físico. Três cenários foram analisados: baseline, degradação de rede e falha de serviços críticos. Os resultados mostram que falhas podem elevar o custo energético por requisição em até 5,6 vezes, evidenciando que a resiliência é essencial não apenas para disponibilidade, mas também para sustentabilidade.*

1. Introdução

A adoção de arquiteturas de microsserviços tem transformado o desenvolvimento de software, ao permitir maior escalabilidade e agilidade. Entretanto, a fragmentação em múltiplos serviços interdependentes introduz desafios significativos na gestão de falhas, especialmente em ambientes de nuvem privada com recursos limitados, onde problemas de rede ou em serviços individuais podem gerar efeitos em cascata e comprometer a disponibilidade do sistema [Gunawi et al. 2016] [Callou and Vieira 2024].

Além dos impactos econômicos do *downtime*, cresce a preocupação com a eficiência energética, uma vez que falhas podem levar ao desperdício de recursos ao processar requisições que não resultam em sucesso [Leonardo and Callou 2025] [de Oliveira and Callou 2025]. Nesse contexto, a Engenharia do Caos surge como uma abordagem para avaliar proativamente a resiliência de sistemas por meio da injeção controlada de falhas. Contudo, ainda são escassos os estudos que analisam conjuntamente os efeitos dessas falhas sobre o desempenho e o consumo energético em ambientes reais. Este trabalho busca preencher essa lacuna ao apresentar uma avaliação experimental em um ambiente de nuvem privada baseado em OpenNebula, utilizando a aplicação TeaStore [Wurster et al. 2018] como Sistema sob Teste. Em síntese, as principais contribuições deste artigo são:

- **Metodologia empírica:** Integração de injeção de falhas (*NetEm* e *Pumba*) com medição energética pontual via RAPL (*Scaphandre*) em *testbed* físico real.
- **Quantificação analítica:** Demonstração de que falhas não apenas reduzem a disponibilidade, mas aumentam drasticamente o custo energético por requisição, revelando ineficiências operacionais importantes que afetam a sustentabilidade da nuvem.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 descreve o ambiente experimental e a metodologia; a Seção 4 discute os resultados; e a Seção 5 conclui o trabalho.

2. Trabalhos Relacionados

A literatura de avaliação de desempenho em sistemas distribuídos é vasta, mas apresenta lacunas significativas no que tange à combinação de resiliência e sustentabilidade em ambientes reais. [Basiri et al. 2016] introduziram formalmente a Engenharia do Caos no contexto da Netflix com o *Chaos Monkey*. Embora o trabalho seja seminal, ele foca em ambientes de nuvem pública em escala massiva e não aborda a mensuração de impacto energético das falhas. A ausência de métricas de energia em seus estudos limita a compreensão do custo ambiental de *outages* parciais, um fator crítico para datacenters modernos que buscam eficiência operacional.

[Wurster et al. 2018] apresentaram o TeaStore, um *benchmark* de microsserviços amplamente utilizado. Vários trabalhos utilizam o TeaStore para avaliar escalabilidade e desempenho, mas a maioria foca em cenários ideais ou de sobrecarga de carga. No entanto, esses trabalhos geralmente utilizam ambientes puramente virtualizados sem acesso a contadores de energia de hardware (RAPL), o que impede a análise da eficiência energética sob estresse.

O presente trabalho se diferencia ao propor uma abordagem puramente empírica em um testbed físico real (servidor Xeon E5), combinando Engenharia do Caos com medição de energia via Scaphandre. Isso permite não apenas validar a resiliência funcional (o sistema funciona?), mas também a eficiência operacional (quanto isso custa em energia?), uma dimensão frequentemente negligenciada na literatura de tolerância a falhas.

Adicionalmente, embora ferramentas mais abrangentes de Engenharia do Caos (como *Litmus Chaos* e *Chaos Mesh*) e de monitoramento energético (como *Kepler*) ofereçam ecossistemas robustos para *clusters Kubernetes*, a escolha pela combinação

direta de *Pumba*, *NetEm* e *Scaphandre* demonstrou-se mais aderente ao contexto deste estudo. Essa abordagem minimiza o *overhead* de orquestração no *OpenNebula* e garante captura direta dos dados via RAPL.

3. Ambiente Experimental e Metodologia

O ambiente experimental é composto por um servidor físico dedicado, denominado nebulosa, que atua como host para a nuvem privada. A escolha de hardware físico dedicado, em detrimento de instâncias na nuvem pública, foi crucial para garantir acesso privilegiado aos contadores RAPL do processador para medição de energia. A Tabela 1 resume a especificação dos componentes principais. O OpenNebula foi escolhido como orquestrador por ser uma solução robusta de código aberto para nuvens privadas, permitindo gerenciar o ciclo de vida das VMs e configurar redes virtuais com facilidade. A rede local garante baixa latência entre a estação de geração de carga e o SUT, isolando a variável de rede externa e focando na degradação injetada internamente.

Tabela 1. Especificação do Ambiente Experimental (PEDAL)

Componente	Especificação
Servidor Físico (nebulosa)	Intel Xeon E5-2689 @ 3.60 GHz (16 threads), 62,21 GiB RAM
Sistema Operacional Host	AlmaLinux 9.7, Kernel 5.14.0-611
Plataforma de Nuvem	OpenNebula 7.0.1 (Gerenciamento de VMs via KVM)
VM SUT (RockyLinux9-Tea)	Rocky Linux 9.6, 2 vCPU, 4 GiB RAM, IP 192.168.1.101
VM Monitoramento	Ubuntu 24.04 LTS, 2 vCPU, 4 GiB RAM, IP 192.168.1.36
Estação Cliente	HP ProBook (Fedora 43), IP 192.168.1.15

O TeaStore [Wurster et al. 2018] foi escolhido por ser um benchmark representativo de e-commerce, composto por 7 serviços interdependentes. Ele é implantado na VM SUT (Rocky Linux 9.6) utilizando Docker Compose. A escolha do Docker se deve à sua ampla adoção na indústria e à facilidade de isolar falhas em contêineres individuais. A VM SUT foi configurada propositalmente com recursos limitados (2 vCPU, 4GB RAM) para executar todos os serviços. Essa contenção de recursos é intencional para estressar o sistema e evidenciar degradações de desempenho que poderiam ser mascaradas em ambientes superdimensionados. A VM de monitoramento centraliza a coleta de dados com o objetivo de evitar o efeito de overhead no SUT. Para tal, utiliza-se o Prometheus e o Grafana. Por fim, o Scaphandre constitui a principal ferramenta para análise energética: instalado diretamente no host físico, ele disponibiliza métricas de consumo permitindo derivar o custo energético por requisição atendida [Hubblo 2022].

A metodologia segue os princípios da Engenharia do Caos. Primeiro, definiu-se o estado estável através de um baseline sem falhas. Em seguida, formulou-se a hipótese de que o sistema recuperaria sua capacidade operacional após a remoção da falha. Foram definidos três cenários de experimentação:

- **Cenário 1 (Baseline):** Estabelece a linha de referência de desempenho e consumo energético em condições ideais.
- **Cenário 2 (Degradação de Rede):** Injeção de falhas via NetEm, adicionando 200 ms de latência e 10% de perda de pacotes na interface da VM SUT [Gill et al. 2011].
- **Cenário 3 (Falha de Contêiner):** Injeção de falhas via Pumba, executando um comando *kill* no contêiner *teastore-auth* para avaliar o tempo médio de recuperação (MTTR).

A geração de carga foi realizada pelo Apache JMeter executado na estação cliente, simulando 10 usuários virtuais. O número reduzido é proposital para competir com a CPU limitada da VM SUT. Para assegurar o rigor estatístico da avaliação e contornar a variabilidade inerente aos sistemas distribuídos, cada cenário experimental foi executado 30 vezes de forma independente. Os dados quantitativos extraídos correspondem às médias dessas execuções, acompanhadas do respectivo desvio padrão.

4. Resultados e Discussão

Os experimentos evidenciaram padrões de degradação de desempenho e ineficiência energética, conforme sintetizado na Tabela 2. No cenário de Baseline, o sistema já operava próximo à saturação (uso de RAM em torno de 80%), o que explica o throughput moderado de 12,4 requisições por segundo e indica limitação de recursos mesmo em condições ideais.

Tabela 2. Síntese dos Resultados Experimentais (Média $\pm$ Desvio Padrão)

Métrica	Cenário 1: Baseline	Cenário 2: Rede	Cenário 3: Falha
Throughput (req/s)	$12,4 \pm 0,8$	$4,1 \pm 0,5$ (-66,9%)	$1,8 \pm 0,3$ (-85,5%)
Latência p50 (ms)	320 ± 15	1.850 ± 120 (+478%)	4.500 ± 210 (+1306%)
Taxa de Erros (%)	$0,8 \pm 0,1$	$18,3 \pm 2,1$	$72,1 \pm 4,5$
Consumo Total (W)	$42,5 \pm 1,2$	$38,2 \pm 1,5$	$35,1 \pm 1,8$
Custo Energ. (J/req)	$3,4 \pm 0,2$	$9,3 \pm 0,8$	$19,2 \pm 1,4$
MTTR (s)	-	38 ± 4	68 ± 7

No Cenário 2 (Degradação de Rede), a introdução de latência e perda de pacotes provocou redução do throughput. Esse fenômeno está diretamente associado ao comportamento do TCP sob perdas, especialmente o mecanismo de *exponential backoff*. Com a falha na rede, pacotes precisam ser retransmitidos sucessivamente, acumulando filas de conexões; a CPU permanece ociosa aguardando operações de I/O, o que eleva a latência sistêmica e reduz drasticamente a eficiência, custando energia sem processamento útil.

No Cenário 3 (Falha no serviço de Autenticação), o impacto foi mais severo, pois a indisponibilidade de um serviço crítico comprometeu grande parte da aplicação. O MTTR de 68 segundos foi severamente dominado pelo tempo de inicialização da *Java Virtual Machine* (JVM). Apesar de o contêiner Docker reiniciar rapidamente, o *overhead* de CPU exigido pelo *Garbage Collection* (GC) inicial e pela compilação *Just-In-Time* (JIT) do Java formam um gargalo restritivo para a resiliência imediata.

Do ponto de vista energético, observa-se que, embora o consumo total (em Watts) tenha diminuído nos cenários de falha, o custo por requisição aumentou significativamente. No Cenário 2, atingiu $9,3\ J/req$, devido a retransmissões de TCP. No Cenário 3, alcançou $19,2\ J/req$, refletindo o pico da JVM e o processamento de requisições malsucedidas (HTTP 500). Esses resultados mostram que falhas não implicam economia de energia; ao contrário, levam a um severo desperdício operacional.

5. Conclusão

Este trabalho apresentou uma avaliação experimental abrangente de tolerância a falhas e resiliência em um ambiente real de nuvem privada. Por meio da adoção de uma abordagem baseada em Engenharia do Caos, integrada à medição de consumo energético via

Scaphandre em um testbed físico, foi possível quantificar não apenas a degradação funcional do sistema, mas também os impactos energéticos. Os resultados evidenciam que falhas de rede e indisponibilidade de serviços críticos podem elevar o custo energético por requisição em até 5,6 vezes.

Embora a carga simulada (10 usuários) seja restrita, ela se justifica pela contenção arquitetural imposta (2 vCPUs) para garantir saturação visível. Da mesma forma, o emprego exclusivo do *TeaStore* permitiu um controle mais estrito do *baseline*. Contudo, reconhecemos que esses fatores limitam a generalização dos achados para cenários massivos. Portanto, como perspectivas futuras, pretende-se avaliar o sistema sob estresse com milhares de usuários virtuais concorrentes, além de aplicar a mesma instrumentação a microsserviços implementados em outras linguagens (como *Go* ou *Node.js*), visando isolar o peso do *Garbage Collection* e provar o modelo em diferentes padrões arquiteturais.

Agradecimentos

Os autores agradecem ao suporte financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) para a realização desta pesquisa.

Referências

Basiri, A., Behnam, N., de Rooij, R., Hochstein, L., Kosewski, L., Reynolds, J., and Rosenthal, C. (2016). Chaos engineering. *IEEE Software*, 33(3):35–41.

Callou, G. and Vieira, M. (2024). Availability and performance analysis of cloud services. In *Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing*, LADC '24, page 262–271, New York, NY, USA. Association for Computing Machinery.

de Oliveira, A. B. and Callou, G. (2025). Quantifying the impact of security strategies on the performance and availability of cloud services. In *2025 IEEE 36th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 225–228.

Gill, P., Jain, N., and Nagappan, N. (2011). Understanding network failures in data centers: measurement, analysis, and implications. *SIGCOMM Comput. Commun. Rev.*, 41(4):350–361.

Gunawi, H. S., Hao, M., Suminto, R. O., Laksono, A., Satria, A. D., Adityatama, J., and Eliazar, K. J. (2016). Why does the cloud stop computing? lessons from hundreds of service outages. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*, pages 1–16.

Hubblo (2022). Scaphandre: Energy consumption metrology agent. Disponível em: `https://github.com/hubblo-org/scaphandre`. Acesso: 06 abr. 2026.

Leonardo, W. and Callou, G. (2025). Avaliação da disponibilidade do serviço nextcloud hospedado em nuvem privada. In *Anais do XXVI Workshop de Testes e Tolerância a Falhas*, pages 85–98, Porto Alegre, RS, Brasil. SBC.

Wurster, M., Mendoza, D. E., Scharr, T., Straesser, M., and Kounev, S. (2018). TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research. In *Proceedings of the 26th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 223–236.