



VESTUÁRIO VIRTUAL: INTEGRAÇÃO DE ROUPAS 3D EM FOTOGRAFIAS

DIEGO MARCHIONI, GABRIEL RAJÃO, LUIZA DIAS,
SOPHIA CARRAZZA



CONTEXTUALIZAÇÃO

Problema: A alta necessidade de rapidez e praticidade do mundo atual exige que escolhamos peças de roupa de forma ágil e eficaz.



Resolução: Aplicação de um guarda-roupa virtual com simulação de um provador 3D das roupas cadastradas pelo usuário, em diferentes combinações.



OBJETIVOS

- Agilidade e praticidade na escolha de roupas
- Organização digital do guarda-roupa
- Simulação realista em 3D
- Otimização do tempo
- Redução da indecisão
- Experiência personalizada



REFERÊNCIAS BIBLIOGRÁFICAS

- **FENGYI, Liu; LIU, Siru.** *3D Garment Design Model Based on Convolution Neural Network and Virtual Reality.* *Computational Intelligence and Neuroscience*, v. 2022, p. 1-12, 2022. DOI: <https://doi.org/10.1155/2022/9187244>.
- **SAITO, Shunsuke; SIMON, Tomas; SARAGIH, Jason; JOO, Hanbyul.** *PIFuHD: Multi-Level Pixel-Aligned Implicit Function for High-Resolution 3D Human Digitization.* *arXiv preprint arXiv:2004.00452*, 2020. Disponível em: <https://arxiv.org/abs/2004.00452>. Acesso em: 8 set. 2025.
- **REHMAN, Abdul.** *Beyond the Surface: Advanced 3D Mesh Generation from 2D Images in Python.* *Medium, Red Buffer*, 16 fev. 2024. Disponível em: <https://medium.com/red-buffer/beyond-the-surface-advanced-3d-mesh-generation-from-2d-images-in-python-0de6dd3944ac>. Acesso em: 8 set. 2025.



PUC Minas

TÉCNICAS E FERRAMENTAS

- Python (backend/modelos)
- React ou Streamlit (frontend)
- OpenCV para o processamento de imagens
- PyTorch DDP (multi-GPU)
- MPI para computação paralela



PUC Minas

METODOLOGIA

2.1. Configuração de Ambiente e Pipeline Inicial

- Preparação do ambiente (Python, PyTorch, Docker)
- Organização e análise de datasets (Viton)

2.2. Projeto e Implementação do Modelo de Try-On

- Treinamento dos modelos TOM e GMM
-

2.3. Refino e Interface de Usuário

- Desenvolvimento de interface gráfica (React),

2.4. Paralelismo, Distribuição e Deploy

- Treinamento paralelo e distribuido,
- Deploy containerizado (Docker)

RESULTADOS ESPERADOS

Simulação realista de peças sobre o corpo do usuário em 3D, melhorando a experiência de experimentação



CRONOGRAMA - SPRINT 2

EST - GABRIEL

SEMANA 1:

- Integrar modelo baseline

SEMANA 2:

- Criar interface visual inicial

SEMANA 3:

- Implementar treino/inferência distribuído com PyTorch DDP (2 GPUs locais/simuladas).

SEMANA 4:

- Implementar DataLoader paralelo (PyTorch num_workers, prefetch_factor).

EST - LUIZA

- Adicionar módulo de detecção de pose
- Otimizar pipeline: reduzir tempo de inferência
- Preparar container Docker com API e modelo
- Aplicar augmentations em paralelo (Albumentations + multiprocessing).

EST - SOPHIA

- Adicionar parsing/segmentação humana
- Implementar upload de imagens pelo usuário
- Testar orquestração simples com Ray Serve ou Triton
- Validar ganho de desempenho em tempo de treinamento/inferência.

EST - DIEGO

- Ajustar warping da roupa
- Testar variante de maior qualidade para modelo baseline.
- Ajustar qualidade final com pós-processamento
- Preparação da documentação e GitHub para a apresentação da Sprint 2

CRONOGRAMA - SPRINT 3

EST - GABRIEL

SEMANA 1:

- Configurar PyTorch DDP em múltiplas GPUs reais (ou simulação via Docker).

SEMANA 2:

- Integrar Ray Serve ou Triton Inference Server para servir modelo em cluster.

SEMANA 3:

- Deploy em cluster Docker/Kubernetes (mesmo local ou cloud).

EST - LUIZA

- Alternativa: integrar Horovod para multi-node.
- Balanceamento de carga entre réplicas do modelo.
- Testes e refinamentos finais

EST - SOPHIA

- Validar escalabilidade: medir speedup vs. 1 GPU.
- Implementar cache de resultados para imagens repetidas.
- Documentação do processo e síntese dos resultados obtidos

EST - DIEGO

- Refinar UI final (upload, galeria, histórico de combinações).
- Pipeline otimizado de ponta a ponta (pré-processamento paralelo + inferência distribuída).
- Preparação da apresentação para a Sprint 3

CONCLUSÕES

- CONSEGUIMOS IMPLEMENTAR O PROCESSAMENTO E ANÁLISE DE IMAGEM COM RESULTADOS SATISFATÓRIOS
- POSSIBILIDADE DE APERFEIÇOAMENTO COM O REFINAMENTO DO MODELO
- CONSIDERAÇÕES INICIAIS PARA O PARALELISMO DE BLOCOS EXECUTADOS PELA CPU



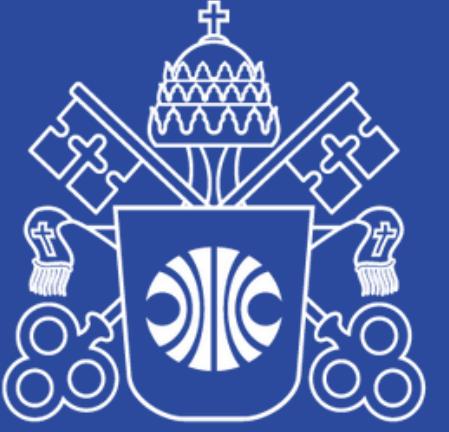
OBRIGADO!



PUC Minas

AUTORES

- THIAGO M COSTA
- DAVI M VASCONCELOS
- CARLOS M ADERALDO
- NABOR C MENDONÇA



PUC Minas

DIFERENÇA DAS LINGUAGENS TESTADAS

C#

- FORTEMENTE TIPADA
- ORIENTADA A OBJETOS
- AMBIENTE DE EXECUÇÃO
BASEADO EM .NET

Java

- FORTEMENTE TIPADA
- ORITENTADA A OBJETOS
- EXECUTADA NA MÁQUINA
VIRTUAL JAVA (JVM)



PUC Minas

PADRÕES TESTADOS

RETRY

O padrão *Retry* consiste em tentar novamente uma operação que falhou, com uma estratégia de retentativas em intervalos de tempo determinados. No estudo, foi observado que o padrão *Retry* é mais interessante em situações onde ocorrem falhas transitórias/curto-prazo

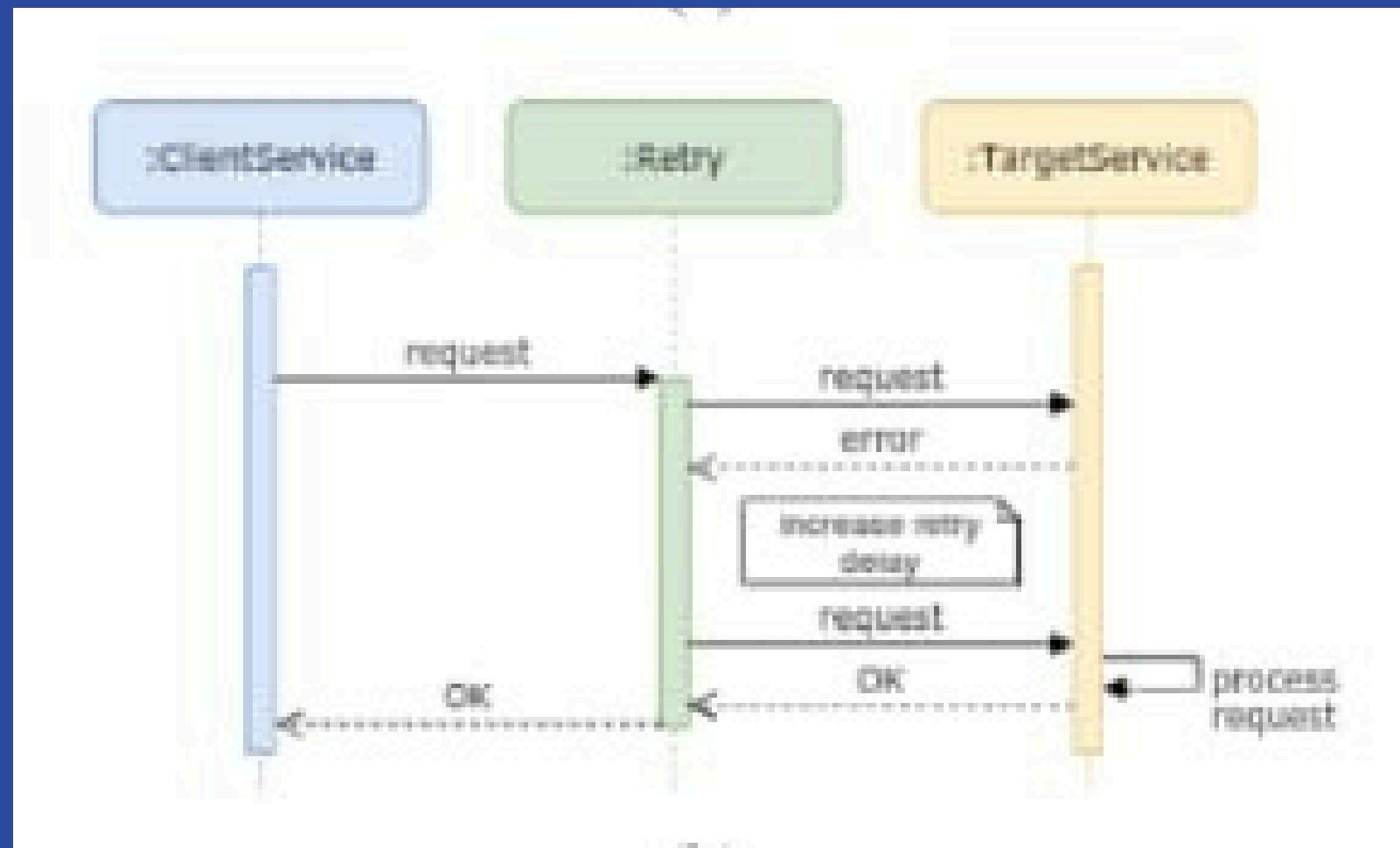
CIRCUIT BREAKER

O padrão *Circuit Breaker* atua como um interruptor que pode abrir e fechar o circuito para o servidor em caso de falhas recorrentes. Quando o circuito está aberto, as requisições não são enviadas ao servidor, evitando sobrecarregá-lo. No estudo, foi observado que o *Circuit Breaker* é mais adequado para situações onde ocorrem falhas de tempo indeterminado/longo-prazo

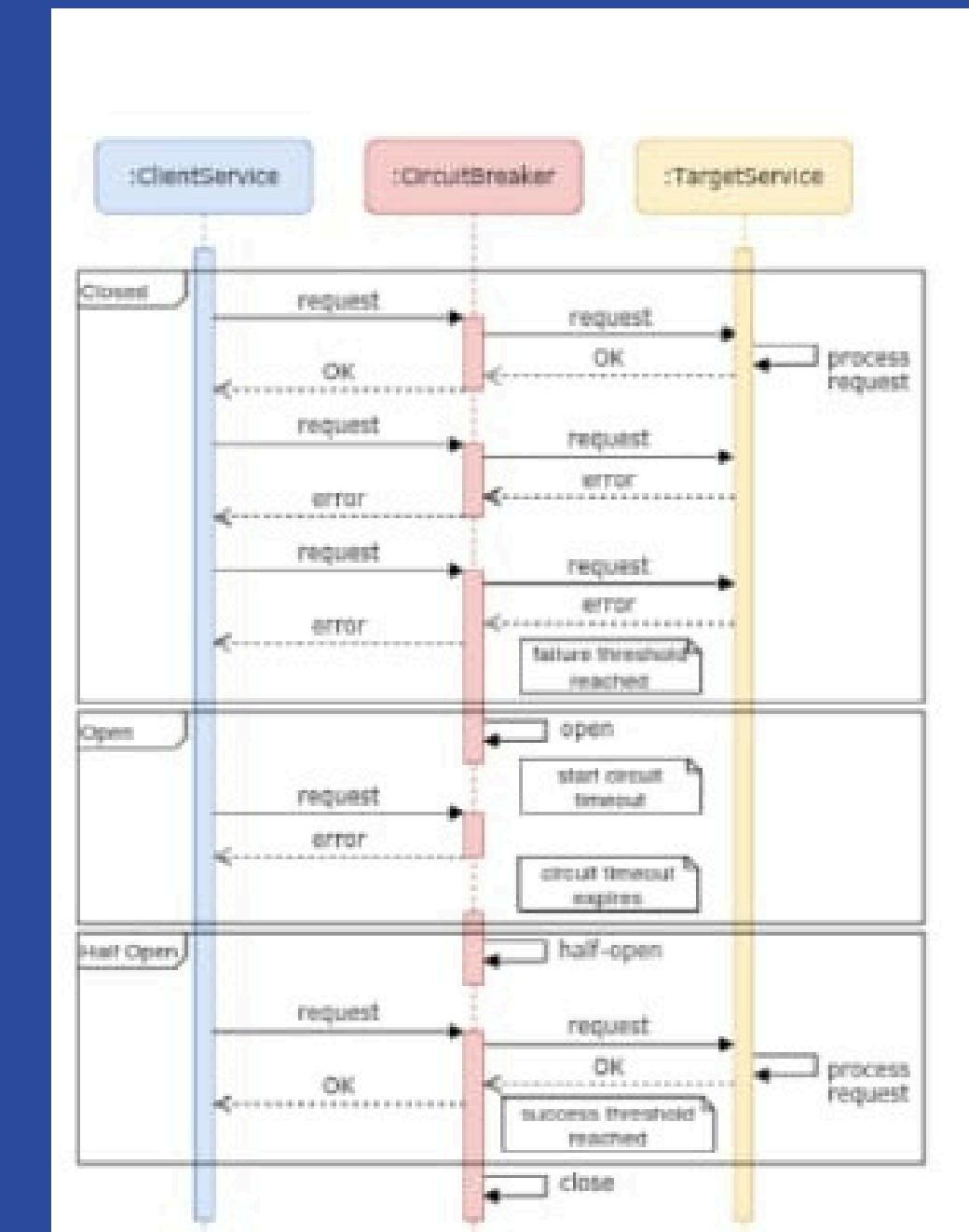


PADRÕES TESTADOS

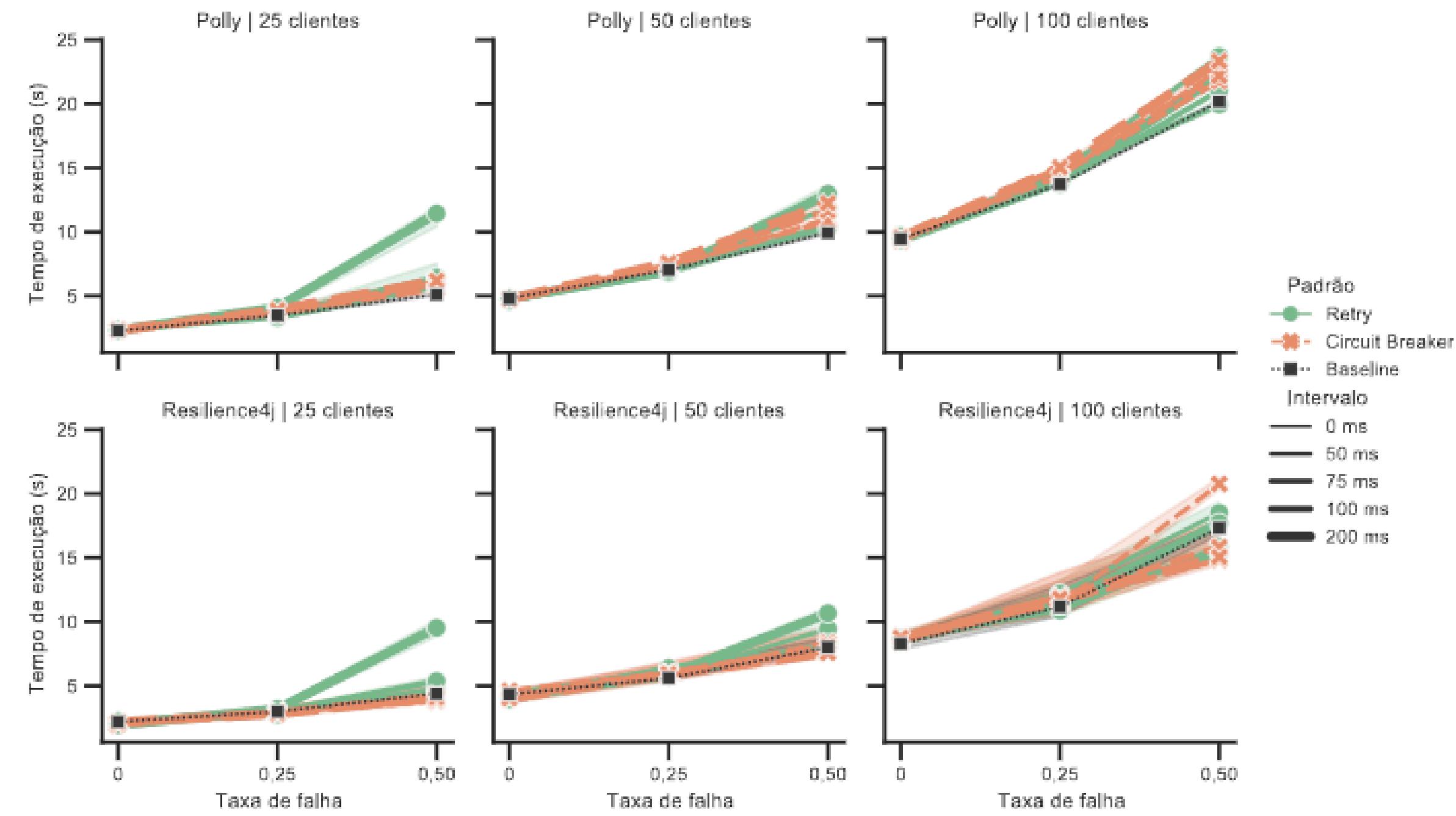
RETRY



CIRCUIT BREAKER



RESULTADOS - TEMPO DE EXECUÇÃO



RESULTADOS - TAXA DE CONTENÇÃO

