Aspecto	Comparison of Concurrency Technologies in Java	Meu Teste
Ambiente de Execução	Duas máquinas físicas (MacBook Pro 2019). Processador Intel Core i7 2,6 GHz (6 núcleos). 16 GB DDR4 de memória. macOS 14.2.1 (64 bits). Conexão Ethernet direta entre as máquinas para eliminar variações de rede.	Uma máquina MacBook Air M2 com 8 GB de memória executou as requisições. Uma máquina Azure Standard E4s v3 (4 vCPUs, 32 GiB de memória) atuou como servidor.
Framework e Linguagem	Spring Boot 3.2 e Java 21.	Spring Boot 3.4.11 e Java 21.
Ferramentas Utilizadas	Vegeta responsável por enviar ataques HTTP controlados e medir throughput e latência. VisualVM e coleta da JVM monitoraram CPU, heap, threads vivas, sincronização e eventos do Garbage Collector.	Foram utilizados dois cenários possíveis: (a) Vegeta para envio de carga controlada e medição de throughput e latência; (b) Apache JMeter como alternativa para geração de requisições e análise de desempenho. Em ambos os casos, o VisualVM foi utilizado para monitorar CPU, heap, threads vivas e eventos do Garbage Collector.
Arquitetura do Teste	Duas máquinas físicas interligadas por conexão Ethernet direta. Uma máquina atuou como cliente (gerando carga via Vegeta) e a outra como servidor (executando a aplicação Java). Foram testadas três tecnologias de concorrência: Platform Threads, Virtual Threads e Reactive Streams.	Duas máquinas conectadas pela internet. O Mac atuou como cliente (realizando as requisições), enquanto a máquina Azure atuou como servidor executando a aplicação Java. Foram testadas duas tecnologias de concorrência: Platform Threads e Virtual Threads.
Configuração de Carga	O teste foi dividido em duas fases principais, ambas com duração de 10 minutos: (1) Fase de aumento gradual de carga (load ramping) — crescimento progressivo da taxa de requisições até atingir o ponto de saturação. (2) Fase de carga constante — manutenção de uma taxa estável de requisições para medir desempenho sob uso sustentado.	O teste utilizou a mesma metodologia: duas fases de 10 minutos cada. (1) Fase de aumento gradual de carga até o limite de saturação. (2) Fase de carga constante para avaliar estabilidade e consumo de recursos.
Comportamento Observado	Platform Threads apresentaram maior consumo de memória e limitação na escalabilidade. Virtual Threads demonstraram melhor aproveitamento da CPU e maior estabilidade sob carga sustentada. Reactive Streams tiveram boa eficiência, porém com maior complexidade de implementação.	Tendência esperada: Platform Threads devem apresentar maior consumo de memória e menor escalabilidade. Virtual Threads devem oferecer melhor uso da CPU e maior estabilidade em cargas prolongadas.
Execuções	5 execuções.	5 execuções.