Nome	Métricas	Benchmark	Warmup	Exec.	Excl.	Resultado	O que comparou	Teste
Comparison of Concurrency Technologies in Java	Latência, taxa de transferência, taxa de amostragem, respostas do servidor, bytes de entrada e saída.	Não Informado	Três execuções de 60s, cada uma com 300 requisições/seg. Uma execução de 2min, usando a taxa real do teste.	5	Outliers	Computation Test (CPU-bound): Threads tradicionais tiveram o melhor desempenho em throughput e latência. I/O Test (I/O-bound): O reactive Framework teve maior escalabilidade, menor uso de CPU e memória. Mixed Test – Computation focused: As threads virtuais tiveram alta taxa de requisições e baixa latência. Mixed Test – I/O focused: As threads virtuais mantiveram desempenho alto, enquanto as tradicionais tiveram alta latência e menor throughput.	Threads tradicionais, virtuais e reactive Framework em teste de I/O-Bound, CPU-Bound e testes mistos	I/O Test – Thread.sleep(100 ms). Computation Test – Multiplica matrizes 200×200. Mixed Test – Multiplica matrizes 200×200 + pausa curta (25 ms). Mixed Test – I/O focused – Multiplica matrizes 50×50 + pausa longa (100 ms).
Avaliação dos mecanismos de concorrência na API do Java 8	Tempo de execução (µs) Consumo de memória (MB)	Não informado	Não	60	Não Houve	Quicksort com ForkJoinPool em máquina com 8 processadores lógicos obteve o melhor resultado em tempo, Quicksort e ForkJoinPool apresentaram os menores tempos na maioria dos cenários.	Os algoritmos Quicksort, Mergesort e Pidgeonholesort com single thread, threads, executorService e fork/Join em dual-core, i5 e i7	CPU-bound

Quadro 0.0.1. Trabalhos relacionados (Parte 1)

Nome	Métricas	Benchmark	Warmup	Exec.	Excl.	Res.	O que comparou	Teste
Uma análise comparativa entre threads e green threads no Java	Tempo de execução (µs)	Biblioteca JMH	1	30	Não Houve	Threads virtuais foram pelo menos 10 vezes mais rápidas em todos os testes	Tempo de instanciar, iniciar, realizar join e troca de contexto em threads tradicionais e virtuais	CPU-bound I/O-bound (simulado): Criou 100.000 threads, cada thread aguarda 100 ms, invoca Thread.yield() e aguarda mais 100 ms.
Benchmarking the Performance of Java Virtual Threads in High-Throughput Workloads	Taxa de transferência (req/s), latência, uso CPU (%) e memória	Não informado	CPU-bound: Usou 10 threads por 5 min para aquecer I/O-bound: 10 usuários concorrentes durante 60s	3	Execuções com erros de rede descartadas	CPU-bound: Threads tradicionais mais rápidas, maior uso de memória, ambas usaram 100% da CPU, resultados semelhantes em latência e throughput I/O-bound: Threads virtuais mais rápidas, menor uso de memória e CPU, throughput maior, latência menor	Threads virtuais e tradicionais em cálculo de números primos e requisições com ação bloqueante	CPU-bound: Calcular números primos em um intervalo I/O-bound: Duas aplicações Java: uma envia requisições (GET), outra simula operação bloqueante.

Quadro 0.0.2. Trabalhos relacionados (Parte 2)