Uma thread é a menor unidade de processamento existente dentro de um processo. Ela é composta por um identificador de thread (ID), um contador de programa, um conjunto de registradores e uma pilha. As threads compartilham entre si sua seção de código, a seção de dados e outros recursos do sistema operacional, como arquivos abertos e sinais. Cada processo pode conter múltiplas threads, o que permite a execução simultânea de diferentes partes de um programa ou de diferentes tarefas [**?**].

A utilização de threads traz diversos benefícios. Em problemas que envolvem tarefas que exigem alto processamento (CPU-bound) a execução paralela em múltiplos núcleos permite reduzir o tempo total de cálculo. Em problemas que dependem de operações de entrada e saída (I/O-bound) as threads possibilitam que a CPU continue processando outras tarefas enquanto aguarda a conclusão dessas operações, evitando ociosidade. Além disso, em sistemas reativos e servidores, o uso de threads permite que sistemas que atendem múltiplos usuários simultaneamente mantenham respostas rápidas e contínuas, mesmo sob alta carga, garantindo desempenho eficiente e melhor experiência ao usuário. Dessa forma, a utilização adequada de threads é usada como estratégia para otimizar recursos computacionais e melhorar a eficiência de sistemas em diferentes contextos.

O relatório State of the Octoverse 2024 [**?**] demonstra que a linguagem de programação Java está entre as cinco linguagens mais utilizadas na plataforma, o que reforça sua importância como uma das principais tecnologias do desenvolvimento de software.

No lançamento da versão 19 do Java, foram introduzidas as threads virtuais, que são uma forma de implementação de Green Threads. Diferente das threads tradicionais, que são gerenciadas diretamente pelo sistema operacional (SO), as Green Threads são controladas pela linguagem ou pela biblioteca de execução. Nesse contexto, as threads virtuais são threads gerenciadas pela Java Virtual Machine (JVM). Elas podem apresentar comportamentos diferentes das threads tradicionais, enquanto o escalonamento das threads tradicionais é realizado pelo SO, determinando quando cada thread é executada, o escalonamento das threads virtuais é feito pela própria JVM.

O uso de threads tradicionais pode gerar overhead e limitar a escalabilidade. As threads virtuais surgem como uma alternativa para contornar essas limitações. Este trabalho investiga como a utilização de threads virtuais pode impactar o desempenho e a escalabilidade de aplicações concorrentes.

O objetivo deste trabalho é analisar as diferenças de desempenho entre threads tradicionais e threads virtuais, avaliando como cada abordagem impacta a execução de aplicações concorrentes.

Para alcançar o objetivo geral deste trabalho, foram definidos os seguintes objetivos específios:

- Implementar protótipos de aplicações concorrentes utilizando ambas as threads;
- Medir e comparar métricas de desempenho;
- Analisar os resultados obtidos, identificando vantagens e limitações.

Este trabalho se justifica pela necessidade de compreender as vantagens e limitações das threads virtuais em comparação às threads tradicionais. Os resultados podem fornecer informações relevantes para auxiliar na escolha da abordagem mais adequada em diferentes cenários de aplicações concorrentes.