Relatório Técnico: Implementação de Threads com Mutex no Processamento de Pedidos

1. O que são Threads?

Threads são unidades básicas de execução dentro de um processo. Elas permitem que múltiplas tarefas sejam executadas de forma concorrente, compartilhando o mesmo espaço de memória, o que melhora a utilização dos recursos do sistema operacional. O uso de threads é essencial para otimizar o desempenho de programas, especialmente em sistemas que lidam com múltiplas operações simultaneamente, como servidores web, bancos de dados e processamento de pedidos.

O processamento de pedidos em sistemas de logística pode ser um gargalo quando executado sequencialmente. Para otimizar esse processo, utilizamos **threads** para paralelizar tarefas, reduzindo o tempo de resposta e melhorando a eficiência do sistema. No entanto, a concorrência pode causar problemas de sincronização, exigindo o uso de **Mutex (Lock)** para evitar inconsistências em dados compartilhados.

2. Como as Threads Resolvem o Problema

No modelo sequencial, cada pedido é processado **um de cada vez**, resultando em tempos de espera elevados. Com o uso de **threads**, as operações são distribuídas entre diferentes unidades de processamento, permitindo que **múltiplos pedidos sejam tratados simultaneamente**. Isso reduz significativamente o tempo total de execução e melhora a escalabilidade do sistema.

3. Algoritmo Implementado

A solução foi desenvolvida em **Python**, utilizando a biblioteca `threading`. O programa simula o processamento de pedidos com quatro etapas:

- **Validação de dados**
- **Conferência de estoque**
- **Cálculo do frete**
- **Emissão da nota fiscal**

Cada pedido passa por essas etapas, e a implementação utiliza **threads** para processar múltiplos pedidos simultaneamente.

Para garantir a sincronização, foi utilizado um **Mutex (`threading.Lock`)**, impedindo que múltiplas threads alterem a variável compartilhada (`pedidos_processados`) simultaneamente, evitando a condição de corrida.

4. Comparativo de Performance

Para avaliar a melhoria, foram comparados os tempos de execução do processamento **sequencial** e **multithread**.Os resultados mostram que a execução com threads é significativamente mais rápida, pois vários pedidos são processados em paralelo.

Execução	Tempo Total (segundos)
Sequencial	21.83
Com Threads	4.79

5. Conclusão

O uso de **threads** e **Mutex** permitiu a otimização do sistema, reduzindo o tempo de execução e garantindo a integridade dos dados compartilhados. Esse tipo de abordagem é essencial para aplicações de alto desempenho e pode ser expandida para sistemas reais com grande volume de pedidos.