**Introdução**

Este trabalho tem como objetivo implementar uma aplicação produtor-consumidor. Aplicações produtor-consumidor se caracterizam por múltiplas threads utilizando os mesmos objetos, de modo que instruções críticas sejam realizadas de forma sincronizada entre as threads.

Criamos uma aplicação chamada Stalker, que realiza dois tipos de tarefas ao mesmo tempo: baixa páginas do site URI Online Judge (produtor) e extrai informações relevantes das páginas (consumidor). A aplicação foi implementada utilizando a linguagem de programação Python.

**Descrição do problema**

O problema do produtor-consumidor consiste em múltiplos processos que compartilham o mesmo buffer. O produtor insere dados no buffer, enquanto o consumidor retira dados do buffer.

Quando múltiplos processos acessam os mesmos dados simultaneamente, existe a possibilidade de ocorrer uma troca de contexto durante a atualização dos dados. Como instruções de escrita em memória podem não ser atômicas, uma eventual troca de contexto poderia fazer com que processos diferentes utilizem cópias diferentes dos mesmos dados, gerando inconsistência nos dados.

Cada linguagem de programação resolve o problema do acesso concorrente de forma diferente. Internamente, as linguagens chamam uma instrução atômica do sistema operacional, durante a qual é garantido que não haverá troca de contexto. Para o programador, isso pode ser abstraído de diferentes formas: mutex, semáforos, métodos sincronizados, etc.

**Descrição das implementações**

A aplicação criada, Stalker, implementa uma solução para o problema do produtor-consumidor e tem como objetivo gerar estatísticas a partir das informações disponíveis no site URI Online Judge. Como o download (produção) e a análise (consumo) das páginas do site podem demorar, a utilização de múltiplas threads para realizar essas tarefas pode tornar a aplicação muito mais rápida.

Até o momento, o consumidor apenas extrai informações gerais dos perfis dos usuários do site. Após a entrega do trabalho, existe a possibilidade de se adicionar novas funcionalidades à aplicação.

O que ele faz com os dados? Salva em um arquivo json. Exemplo de um arquivo ou dados coletados?

Como exatamente o produtor faz o download da aplicação Que método ele usa? (http) O que ele usa para sincronizar? (Fila com locks)? De onde ele tira o que ele tem que fazer download (arquivo com url)?

Como o consumidor extrai as informações?

Comparação entre o método sequencial e multithread?

Quais são as conclusões? Gagalo da rede, desempenho pode mudar depemdemdp da velocidade da rede, etc

Por que apenas um consumidor consegue atender toda a demanda dos produtores?

Diagrama mostrando uma visão geral da arquitetura do sistema?

Tabela com os testes feitos?

Esse modelo produtor-consumidor é recomendado para aplicações como web mining?

Complemento do objetivo da aplicação? Dados que serão que poderão ser usados por outra aplicação, etc

Mostrar trechos dos codigos que mostra o produtor e o consumidor? Talvez um pseudo código?

Melhorar a descriçãó do que um produtor-consumidor, focando em que tipo de apliacaçãio esse modelo é interessante? Destacar que nessa aplicação o produtor consumidor pode ser uma boa forma de abstração?

Especificar que tipo de mecanismo de proteção está sendo usado para proteger a seção crítica do código? Colorcar um trecho do código onde está a seção crítica?

Colocar referências? Que tipo de referencia? Da documentação do python sobre a queue? Uma que fala sobre web mining?

Caracterizar o tipo de aplicação, destancando que o gargalo está na rede?