Atividade S6

Rafael Sartori M. Santos, 186154

10 de abril de 2020

Avaliação

Esta questão pretende avaliar as diferentes formas de *multithreading* para aplicações em *user-level*.

Enunciado do problema

Quais são os principais possíveis modelos para se implementar *multithreading* do ponto de vista de uma aplicação em nível de usuário? Quais os benefícios de cada abordagem?

Resposta

Há algumas abordagens, as principais são:

- **Implementar *multithreading* completamente em *user-level*:** as aplicações podem criar estruturas para fazer o mesmo trabalho do *kernel*, porém sendo executado em *user-level*. Isso permite executar código concorrente (mesmo que não simultaneamente) em máquinas que não suportam *threads*. Há, no entanto, perda de performance. Como todas as *threads* simuladas executam em um único processo, o *scheduler* do sistema não vê esse tipo de mecanismo, podendo limitar a performance do programa.
- **Implementar *multithreading* utilizando somente a implementação do *kernel*:** aplicações podem utilizar chamadas de sistema para requisitar a criação de uma nova *thread* para cada carga de trabalho, que será tratada pelo *kernel* do sistema, independente da aplicação. O *scheduler* do sistema distribuirá de forma mais otimizada a todas as aplicações o tempo do processador, evitando que aplicações que simulam 2 ou mais *threads* em *user-level* recebam o mesmo *time slice* de uma aplicação que executa código linearmente.
- **Implementar *multithreading* com auxílio do *kernel*:** hoje é comum utilizar *threads* providas pelo *kernel* do sistema, porém de maneira a aproveitar melhor os diversos núcleos do processador, através de complexas estruturas. Fazem isso através de uma *thread* de longa vida para cada núcleo, diluindo o custo de fazer chamadas ao sistema no tempo em que ficam vivas. As *threads* recebem, através de estruturas de dados concorrentes, os trabalhos que precisam executar, aumentando o *throughtput* e diminuindo o tempo gasto em *kernel-level*. É comum em aplicações pesadas, como navegadores, *video encoding/decoding* e servidores de alta demanda de CPU, como os de jogos.