AULA 4(19-11): THREADS, CONCORRÊNCIA E PARALELISMO

A aula foi dividida em duas partes. Na primeira parte, foi apresentado e discutido o conceito de threads, bem como sua definição e uso em um sistema operacional. A segunda parte focou na discussão do vídeo "*Threads On Multicore Systems"*(Threads On Multicore Systems) do canal Core Dumped. Abaixo seguem as principais características de threads:

Definição: threads são unidades de execução dentro de um processo que permitem que partes independentes de um programa rodem simultaneamente. O vídeo destaca que elas surgem como forma de dividir o trabalho em pequenas tarefas, permitindo que a aplicação responda mais rápido ou realize cálculos em paralelo. A ideia central é possibilitar múltiplas linhas de execução convivendo dentro do mesmo espaço de memória, compartilhando dados e recursos. Isso reduz o overhead em comparação com processos separados e abre caminho para aplicações mais dinâmicas e eficientes.

Threads e multicore: Com a popularização de CPUs multicore, threads se tornaram fundamentais para aproveitar o hardware moderno. Cada thread pode ser atribuída a um núcleo diferente, permitindo execução real paralela, não apenas concorrência simulada. O vídeo explica que isso aumenta drasticamente o potencial de desempenho quando tarefas podem ser divididas. Ao distribuir trabalho entre vários núcleos, o sistema reduz o tempo total de execução e melhora a capacidade de lidar com múltiplas tarefas simultâneas. Assim, threads são o principal mecanismo de uso efetivo do paralelismo físico.

Concorrência e paralelismo: threads são a base dos modelos de concorrência e paralelismo em sistemas modernos. Concorrência refere-se à capacidade do programa de lidar com várias tarefas ao mesmo tempo, mesmo que não sejam literalmente executadas simultaneamente. Paralelismo, por sua vez, depende de múltiplos núcleos que possibilitam a execução real em paralelo. Threads podem atuar nos dois cenários, garantindo eficiência tanto em máquinas simples quanto em hardware multicore. Entender essa distinção ajuda a escolher a melhor estratégia de organização do trabalho em um programa.

Sincronização e memória compartilhada: como threads compartilham o mesmo espaço de memória, surge a necessidade de controlar cuidadosamente o acesso aos dados. Sem sincronização adequada, podem ocorrer problemas como condições de corrida e inconsistências lógicas, resultando em comportamentos imprevisíveis. Técnicas como mutexes, semáforos, monitores e barreiras são utilizadas para garantir acesso seguro a recursos críticos. A sincronização, apesar de essencial, adiciona complexidade ao código. Por isso, projetar bem a comunicação entre threads é crucial para evitar erros difíceis de reproduzir.

Desvantagens do uso de threads: O uso de threads, embora vantajoso, traz desafios significativos. É fácil criar situações de deadlock, nas quais duas ou mais threads esperam indefinidamente por recursos bloqueados. Starvation também pode ocorrer quando uma

thread não recebe tempo de CPU suficiente. O gerenciamento incorreto da memória compartilhada causa falhas sutis, geralmente difíceis de depurar. Além disso, criar threads em excesso pode gerar overhead, reduzindo o desempenho ao invés de melhorá-lo. Por isso, boas práticas e planejamento são fundamentais para um sistema multithread eficiente.

CONCLUSÃO

Em conclusão, threads representam um dos pilares fundamentais da computação moderna, permitindo que programas realizem múltiplas tarefas de forma eficiente, responsiva e escalável. Seu uso adequado possibilita melhor aproveitamento do hardware, especialmente em sistemas multicore, ao mesmo tempo em que oferece flexibilidade para organizar o fluxo de execução. No entanto, seu poder vem acompanhado de desafios importantes, como sincronização, gerenciamento da memória compartilhada e prevenção de problemas de concorrência. Quando aplicadas com planejamento e boas práticas, threads elevam significativamente a qualidade e o desempenho das aplicações, tornando-se um recurso indispensável no desenvolvimento de software contemporâneo.

Aluno: João Victor Oliveira
Matrícula: 20240008468