**1. Programação Multithread**

**1.1. O que são Threads?**

Thread é uma forma de um processo dividir a si mesmo em duas ou mais tarefas, podendo executar elas concorrentemente. O suporte a threads é oferecido pelos Sistemas Operacionais, ou por bibliotecas de algumas linguagens de programação.

**1.2. O que são Aplicações Multithreads?**

Um programa single-thread (thread única) inicia na etapa 1 e continua seqüencialmente (etapa 2, etapa 3, o passo 4) até atingir a etapa final. Aplicações multithread permitem que você execute várias threads ao mesmo tempo, cada uma executando um passo por exemplo.

Cada thread é executada em seu próprio processo, então, teoricamente, podemos executar o passo 1 em uma thread e, ao mesmo tempo executar o passo 2 em outra thread e assim por diante. Isso significa que a etapa 1, etapa 2, etapa 3 e etapa 4 podem ser executadas simultaneamente.

Na teoria, se todos os quatro passos durassem o mesmo tempo para executar, poderíamos terminar o nosso programa em um quarto do tempo que levaria para executar os quatro passos sequencialmente.

Então, por que todos os programas não são multithread?

Uma das razões nos diz que junto com a velocidade vem a complexidade. Controlar um grande número de threads requer bastante trabalho e conhecimento. Imaginemos se um passo de alguma forma dependesse das informações no passo 2. O programa pode não funcionar corretamente se o passo 1 termina cálculo antes do passo 2 ou vice-versa.

Para ajudar a entender essa complexidade, vamos fazer uma analogia comparando um programa multithread com o corpo humano. Cada um dos órgãos do corpo (coração, pulmões, fígado, cérebro) são processos e cada processo é executado simultaneamente. Assim, o corpo humano seria como uma grande aplicação multithread. Os órgãos são os processos em execução simultânea, eles dependem uns dos outros, sendo que todos os processos se comunicam através de sinais nervosos e fluxo de sangue, desencadeando uma série de reações químicas. Como em todos os aplicativos multithread, o corpo humano é muito complexo. Se algum processo não obtiver informações de outros processos, ou um processo retardar ou acelerar, vamos acabar com um problema médico. É por isso que, esses processos precisam ser sincronizados corretamente para funcionar normalmente.

**1.3. Quando Utilizar Multithread?**

Quando precisamos de desempenho e eficiência. Por exemplo, vamos pensar em um jogo de vídeo game. Imaginemos um jogo possuindo uma única thread, tanto para o processamento de imagens quanto para o processamento de áudio. Seria possível? Sim, seria. Mas esse é o cenário ideal para garantir a performance? Não. O ideal seria criar threads para o processamento das rotinas de imagens e outra para rotinas de áudio.

Outro cenário comum onde precisaríamos de Multithread seria um sistema de tratamento de mensagens. Imaginemos um aplicativo que captura milhares de mensagens simultaneamente. Não podemos capturar eficientemente uma série de mensagens ao mesmo tempo em que estamos fazendo algum outro processamento pesado, porque senão poderíamos perder mensagens. Então cabe aqui dividir o processamento de captura e processamentos paralelos em threads diferentes.

Cada um destes cenários são usos comuns para multithread e se utilizado de maneira correta, essa técnica vai melhorar significativamente o desempenho de aplicações similares.

**1.4. Quando não utilizar Multithread?**

Geralmente quando aprendemos os recursos de multithread, ficamos fascinados com todas as possibilidades que essa técnica nos oferece e saímos usando–a em todos os tipos de programas que vemos pela frente.

É preciso tomar muito cuidado com isso, pois ao contrário de um programa single thread, estamos lidando com muitos processos ao mesmo tempo, com múltiplas variáveis dependentes. Isso pode se tornar muito complicado de se controlar. Pensemos em multithread, um malabarismo. Malabarismos com uma única bola em nossa mão é bastante simples. No entanto, se somos desafiados a colocar duas dessas bolas no ar, a tarefa é um pouco mais difícil. Imaginemos três, quatro, cinco bolas então, e as coisas vão se tornar cada vez mais difíceis. Como a contagem de bola aumenta temos chances cada vez maiores de deixar cair alguma bola.

Trabalhar com multithreads requer muito trabalho e concentração. É preciso ressaltar que o uso dessa técnica em sistemas simples, geralmente não nos leva a um cenário muito proveitoso. Mesmo que o programador seja um profissional muito bom e desenhe direitinho o sistema utilizando threads, fatalmente o ganho em performance não irá compensar caso seja necessário dar manutenções futuras no sistema.