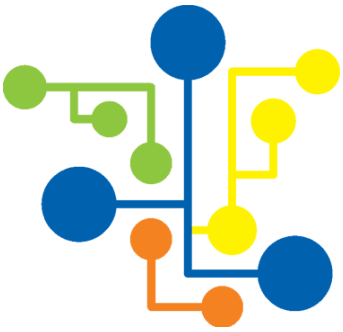


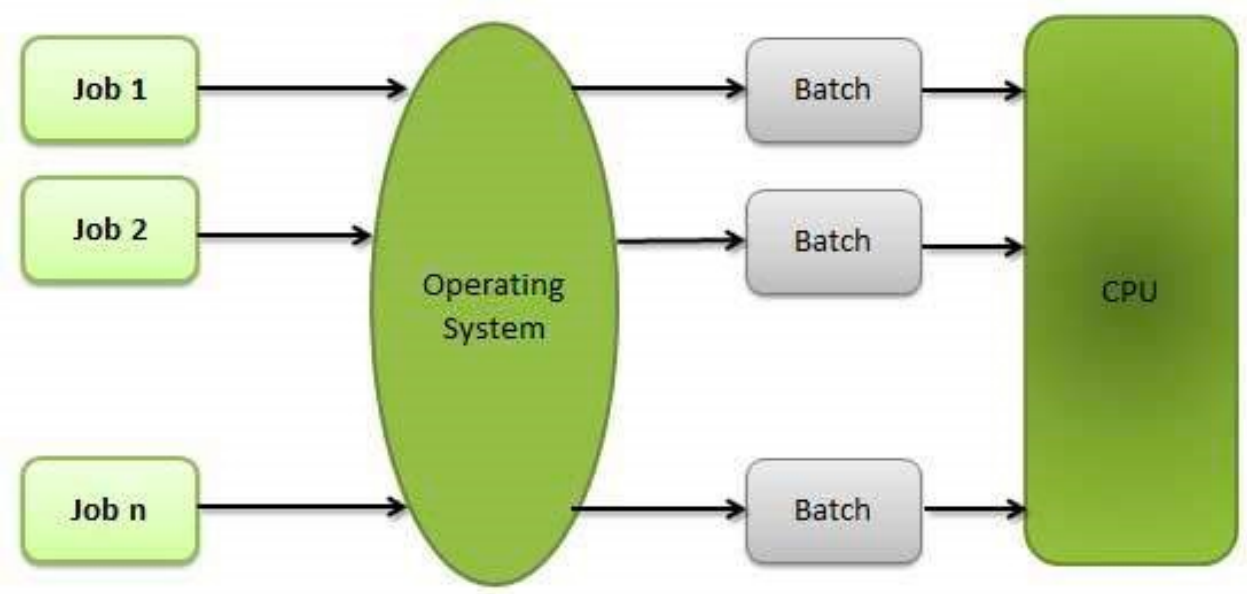
Data Science Academy

www.datascienceacademy.com.br

Machine Learning e IA em Ambientes Distribuídos

O que são Threads?

Uma das palavras mais misteriosas para quem está iniciando em programação paralela é “thread”. No que se refere a hardware, essa palavra ganhou popularidade, quando surgiram os primeiros modelos de processador com múltiplos núcleos. Em princípio era fácil compreender que um dual-core tinha dois núcleos. Entretanto, com a evolução das arquiteturas nas CPUs, surgiu o suporte para múltiplos threads (multithreading). E é aí que muitas pessoas se perguntaram o que realmente mudava. Afinal, o que é essa palavra? Faz diferença um processador trabalhar com o dobro de threads? Esse conceito é importante para a continuidade do curso, então vejamos em detalhes.



Antes de falarmos exatamente sobre as threads, precisamos entender como os processadores e os sistemas operacionais trabalham com os aplicativos. Basicamente, a execução de um programa ocorre, em um primeiro instante, em uma ação do sistema operacional. Quando o usuário abre um aplicativo, o sistema operacional interpreta a ação e requisita que os arquivos relacionados a esse software sejam executados. Claro que qualquer atividade do sistema operacional está sujeita à operação do processador. Todavia, antes que um programa esteja aberto e realmente requisite o trabalho em massa da CPU, ele é apenas carregado na memória RAM, o que não exige uma atividade do processador. Ao efetuar o

carregamento de um programa, o sistema operacional trabalha com processos. Cada software possui um processo (alguns utilizam árvores de processos), cada qual com respectivas instruções para o processador saber como proceder na hora de efetuar os cálculos.

Os chamados “processos” são módulos executáveis, os quais contêm linhas de código para que a execução do programa seja realizada apropriadamente. Isso quer dizer que o processo é uma lista de instruções, a qual informa ao processador que passos devem ser executados e em quais momentos isso acontece. Os processadores trabalham muito bem com os processos, mas a execução de muitos processos simultaneamente acarreta a lentidão da CPU. Isso ocorre porque, mesmo um processador tendo dois ou mais núcleos, existe um limite para ele. Uma CPU com dois núcleos, por exemplo, pode trabalhar com dois processos simultaneamente. No entanto, se você pressionar as teclas “Ctrl + Shift + Esc” no Windows, vai verificar que o sistema operacional trabalha com dezenas de processos ao mesmo tempo. No entanto, tudo parece rodar perfeitamente na sua tela.

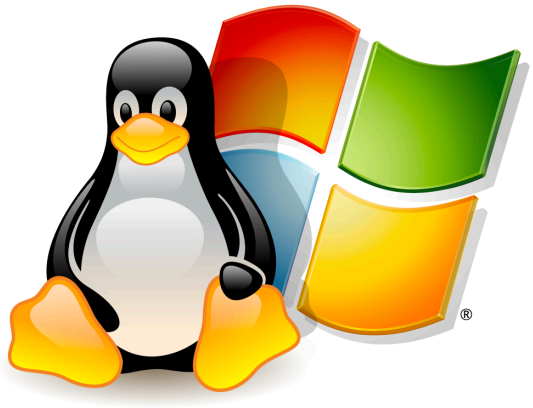
Explicar isso é bem simples. Suponha que estamos tratando de uma CPU com dois núcleos. Em teoria, ela é capaz de executar dois programas ao mesmo tempo. Contudo, você está com seis programas abertos e todos respondendo em tempo real. O processador consegue trabalhar com todos os aplicativos e apresentar resultados satisfatórios devido à velocidade de processamento. Sendo assim, “parece” que os processos são executados simultaneamente.

A princípio, a presença de múltiplos núcleos era suficiente para a maioria dos usuários. Todavia, a evolução dos softwares e dos componentes de hardware requisitou uma divisão ainda melhor das tarefas. As linhas de instruções dos processos adquiriram características únicas, que possibilitaram separá-las para execuções em diferentes núcleos.

Essas linhas de instruções ficaram conhecidas como threads, mas muita gente preferiu traduzir a palavra “thread” para tarefa. A questão é que o nome em si não faz diferença, visto que, de certa maneira, uma linha de instrução é uma tarefa que o processador deverá realizar.

A thread é, portanto, uma divisão do processo principal de um programa. Todavia, nem todos os processos são divididos em múltiplas threads, assim como nem todos os processadores são capazes de trabalhar “tranquilamente” com uma enormidade de threads. Thread é um pequeno programa que trabalha como um subsistema, sendo uma forma de um processo se autodividir em duas ou mais tarefas. É o termo em inglês para Linha ou Encadeamento de Execução. Essas tarefas múltiplas podem ser executadas simultaneamente para rodar mais rápido do que um programa em um único bloco ou praticamente juntas, mas que são tão rápidas que parecem estar trabalhando em conjunto ao mesmo tempo. As diversas threads que existem em um programa podem trocar dados e informações entre si e compartilhar os mesmos recursos do sistema, incluindo o mesmo espaço de memória. Assim, um usuário pode utilizar uma funcionalidade do sistema enquanto outras linhas de execução estão trabalhando e realizando outros cálculos e operações.

Devido à maneira rápida que a mudança de uma thread e outra acontece, aparentemente é como se elas estivessem sendo executadas paralelamente de maneira simultânea em hardwares equipados com apenas uma CPU. Esses sistemas são chamados de monothread. Já para os hardwares que possuem mais de uma CPU, as threads são realmente feitas de forma concorrente e recebem o nome de multithread.



Os sistemas operacionais executam de maneiras diferentes os processos e threads. No caso do Windows, ele trabalha com maior facilidade para gerenciar programas com apenas um processo e diversas threads do que quando gerencia vários processos e poucas threads. Isso acontece porque no sistema da Microsoft a demora para criar um processo e alterná-los é um esforço muito grande. Enquanto isso, o Linux e os demais sistemas baseados no Unix podem criar novos processos de maneira muito mais rápida e executam aplicações de forma muito mais rápida. Ao serem alterados, os programas podem apresentar o mesmo desempenho tanto no Linux quanto no Windows.

Referência:

Programming Massively Parallel Processors: A Hands-on Approach
https://www.amazon.com.br/Programming-Massively-Parallel-Processors-Hands-ebook/dp/B01NCENHQQ/ref=sr_1_1?ie=UTF8&qid=1496248368&sr=8-1&keywords=parallel+programming