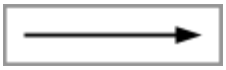
Trabalho 01

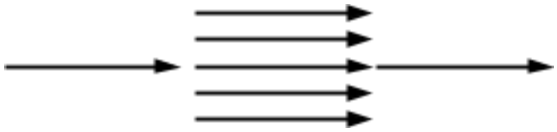
1. Explique as diferenças e semelhanças entre programas sequenciais, concorrentes, paralelos e distribuídos. Em particular, enfatize em que diferem as noções de paralelismo e concorrência. Use exemplos.

O texto a seguir visa explicar do que se tratam programas sequenciais, concorrentes, paralelos e distribuídos. A diferença entre o sequencial e os demais é que o sequencial tem seu funcionamento realizado em uma única linha de execução, sendo seu conjunto de instruções operados em sequência por um único processador. Já os paralelo, concorrente e distribuído tem entre si a semelhança de realizar mais de uma operação simultaneamente.



Sequencial

Programas paralelos são aqueles em que determinadas tarefas podem ser executadas ao mesmo tempo que outras. Esse tipo de recurso é possível por conta dos recursos de hardware disponíveis atualmente, com a presença de várias unidades de processamento em uma única máquina, permitindo múltiplos fluxos de instruções simultâneas. Exemplo desse tipo de programa é a *renderização* de cenas 3D. Partes diferentes da cena podem ser desenhadas por diferentes unidades de processamento, fazendo com que o tempo total para desenho da cena completa seja menor do que se fosse trabalhado sequencialmente. E essa é a ideia por trás do paralelismo de atividades, obter melhor performance na execução de tarefas.



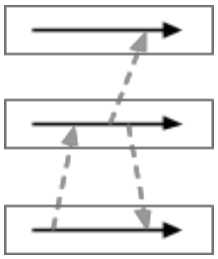
Paralelo

Sistemas concorrentes são aqueles que várias partes diferentes do sistema são executadas simultaneamente, havendo compartilhamento de recursos. Diferente do programa paralelo, o objetivo nesse tipo de sistema não é obter maior performance na execução de uma tarefa, e sim cumprir requisitos de concorrência. Um exemplo de sistema concorrente são os banco de dados. Várias consultas ou outras operações podem ser executadas no mesmo espaço de tempo e haver necessidade de compartilhamento entre os mesmos recursos em mais de uma delas.

É importante diferenciar os conceitos de paralelismo e concorrência. Um sistema de *multithreading*, por exemplo, pode dar a impressão de que várias tarefas são executadas ao mesmo tempo, o que poderia ser confundido com uma execução paralela. Entretanto, essas tarefas podem estar sendo divididas em pedaços de instruções que compartilham o processador com outras tarefas.

Concorrente

Em sistemas distribuídos várias partes do sistema são executados simultaneamente e separados através de uma rede de comunicação, podendo ou não haver compartilhamento de recursos. No caso, várias máquinas diferentes podem realizar parte de uma determinada tarefa. Exemplo desse tipo de sistema é o conhecido ataque *DDOS* (negação de serviço distribuída, do inglês *Distributed Denial of Service*). Nesse tipo de ataque várias máquinas diferentes (inclusive geograficamente separadas) executam requisições ao mesmo tempo para um determinado servidor com o objetivo de fazer com que tal serviço pare de funcionar. Outro exemplo de aplicação de sistema distribuído é para a realização de tarefas que exigem um alto poder de processamento, como decodificação genética.



Distribuído

2. Você já precisou construir programas paralelos, seja por motivos de estudo, seja por motivos profissionais? Escolha o mais complexo desses programas e descreva-o. Explique porque ele precisa realizar várias atividades ao mesmo tempo e em que consistiam essas atividades. Esse programa era "embaraçosamente" paralelo ou exigia sincronização entre as tarefas? Que problemas você enfrentou ao construi-lo (ou ajudar a construi-lo)? Caso você nunca tenha construído um programa assim, procure na Internet um sistema com essas características e examine seu histórico de desenvolvimento, seus bug reports e mensagens de commit, para tentar aprender algo sobre ele.

Foi realizado o desenvolvimento de um sistema para obter os dados de um site com vários pontos de surf (*surf spots* ou "picos") ao redor do mundo. Os dados precisavam ser estruturados de forma que pudessem ser armazenados em um banco de dados, o que inviabilizava o uso de um *web crawler* tradicional. O sistema poderia ser executado de forma síncrona. Entretanto, para acelerar o tempo de obtenção dos dados, a requisição HTTP para cada página e a retirada de informação foi realizada em paralelo.

O sistema não exigia sincronização de tarefas em sua linha principal, uma vez que os dados de cada página não dependiam dos obtidos em uma página diferente. Entretanto, um dos problemas enfrentados estava relacionado à perda de alguma página específica por alguns fatores como erros na interpretação dos dados ou tempo limite de conexão excedido. Esse tipo de problema foi resolvido permitindo se enviar listas de páginas perdidas em execuções posteriores. E para isso, páginas perdidas eram gravadas para serem obtidas depois. Esse processo exigia escrita em arquivo, que era realizada de forma que apenas uma *thread* poderia acessar essa funcionalidade por vez.

Um outro problema enfrentado estava relacionado à quantidade de *thread*s a serem executadas em paralelo. Um numero excessivo de *thread*s sendo executadas levavam a uma situação de execução semelhante ou até mais lenta que executando assíncronamente. Com isso, a quantidade de *threads* foi fixada em um número que permitia um tempo de execução satisfatório.