Arquiteturas Paralelas e Distribuídas

Conceitos básicos de concorrência

Eduardo Furlan Miranda

Baseado em: SOBRAL, JBM. Introdução à Programação Paralela e Distribuída. UFSC, 2009.

Concorrência

- Um programa “ordinário” consiste de declarações de dados e instruções executáveis em uma linguagem de programação

- Programação Concorrente: Execução simultânea de múltiplas partes de um programa, explorando paralelismo para aumentar a eficiência e a responsividade

Programação Concorrente

- As instruções são executadas sequencialmente sobre um computador, o qual aloca memória para reter os dados do programa

- Um programa concorrente é um conjunto de programas sequenciais ordinários os quais são executados em uma abstração de paralelismo

Programação Concorrente

- Usamos a palavra processo para programas sequenciais e reservamos a palavra programa para o conjunto de processos

- Concorrência se preocupa com a estrutura de uma aplicação, enquanto paralelismo se preocupa com a sua execução

Pseudo-Paralelismo

- Um programa concorrente é executado compartilhando-se o poder de processamento de um único processador entre os processos desse programa.
- Em um único processador, a execução dos processos é intercalada, criando a ilusão de paralelismo.
- O sistema operacional gerencia essa alternância, dando a cada processo uma fatia de tempo do processador.
- Objetivo: Aumentar a responsividade do sistema e otimizar o uso dos recursos, mesmo com um único núcleo.

Abstração para Concorrência

- O paralelismo é abstrato porque não requeremos que um processador físico seja usado para executar cada processo.
- A concorrência define a estrutura do programa para lidar com múltiplas tarefas, independentemente da execução física simultânea.
- A abstração permite que um programa concorrente seja executado em diversas arquiteturas, com um ou múltiplos processadores.
- Essa abstração simplifica o desenvolvimento e aumenta a portabilidade do código concorrente.

Exemplos de Concorrência

- Sobreposição de I/O e Computação (Overlapped I/O and Computation): Executar operações de entrada/saída (I/O) simultaneamente com o processamento da CPU, evitando ociosidade.

- Multiprogramação (Multi-programming): Manter múltiplos programas carregados na memória principal, alternando a execução entre eles para maximizar o uso da CPU.

- Multitarefação (Multi-Tasking): Permitir que um único usuário execute múltiplos programas ou tarefas simultaneamente, com o sistema operacional gerenciando a alternância entre eles.

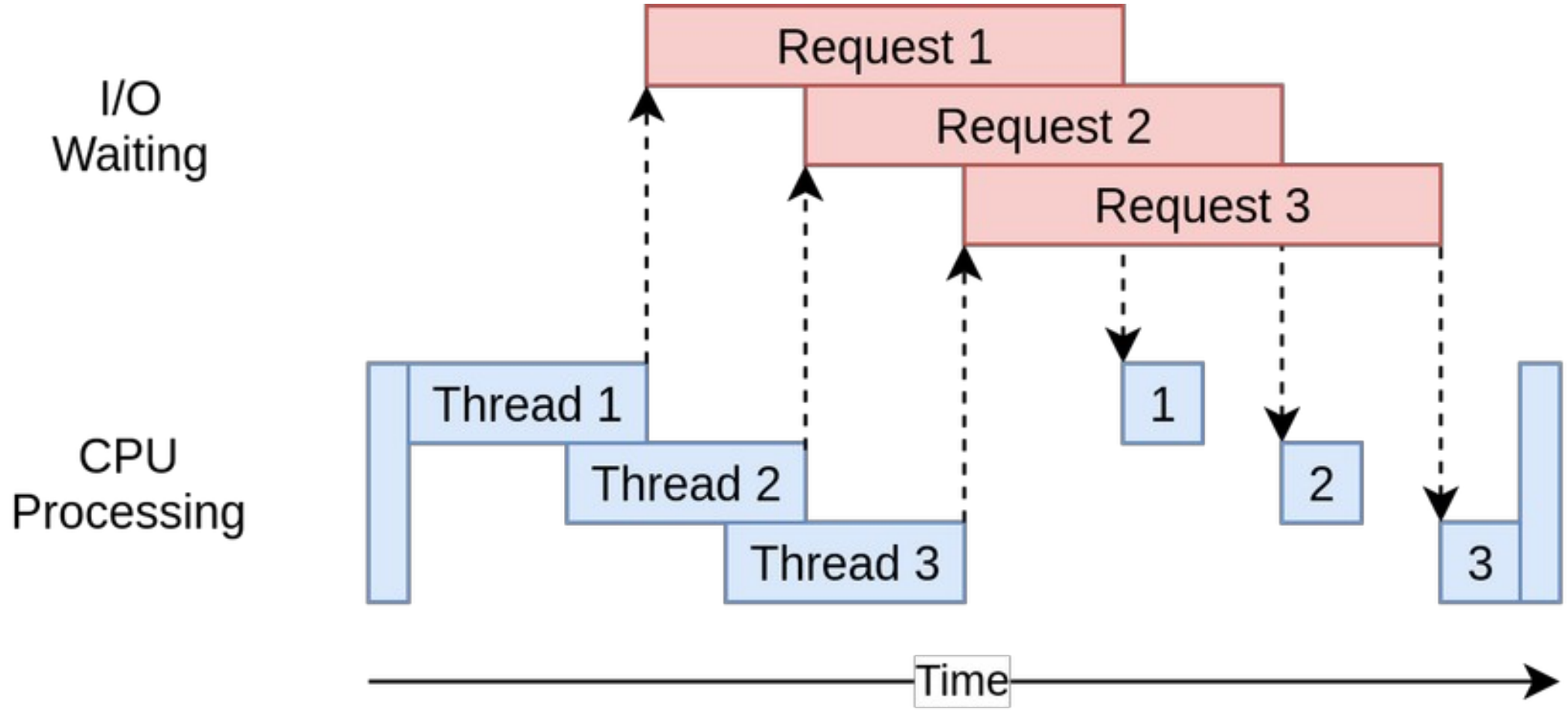
- Essas técnicas melhoram a eficiência e a responsividade do sistema, otimizando o uso dos recursos.

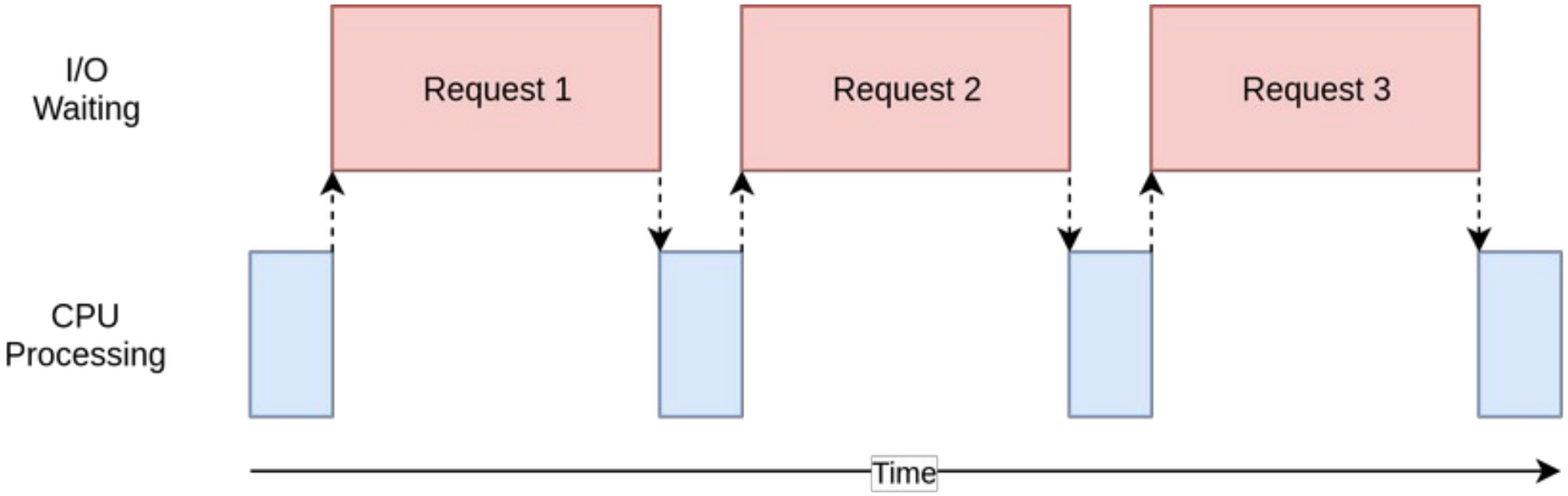
Sobreposição de I/O e Computação

- Em um único processador, o controle de I/O não pode ser feito em paralelo com outra computação.
- No entanto, é possível executar essas tarefas de forma concorrente, intercalando a computação principal com breves momentos dedicados ao controle de I/O.
- O sistema operacional utiliza interrupções (interrupts) para sinalizar a necessidade de atender a uma operação de I/O, interrompendo momentaneamente a execução do processo em andamento.
- Essa técnica melhora a eficiência, evitando que o processador fique ocioso enquanto aguarda a conclusão de operações de I/O lentas.

Concorrência: Acelerando I/O e Computação

- Operações de Entrada/Saída (I/O) lentas causam ociosidade do processador
- Concorrência: executar I/O e computação paralelamente usando processos/threads separados
- Maior eficiência e responsividade do sistema
  - Ex: Salvamento automático em editores de texto


I/O
Waiting
CPU
Processing
Request 1
Request 2
Request 3
Thread 1
Thread 2
Thread 3
1
2
3
Time


I/O
Waiting
CPU
Processing
Request 1
Request 2
Request 3
Time

Execução Concorrente

- Multiprogramação
  - Execução concorrente de múltiplos programas/processos em um processador, maximizando o uso da CPU ao sobrepor I/O e computação de diferentes programas
- Programação Paralela
  - Execução simultânea de partes de um único programa em múltiplos processadores/núcleos, visando reduzir o tempo de execução
- Programação Distribuída
  - Execução de um programa em múltiplos computadores interconectados (uma rede), permitindo a solução de problemas complexos e o processamento de grandes volumes de dados

Time-Slicing: Compartilhando o Processador

- Técnica que permite o compartilhamento do processador entre múltiplos processos,
  - criando a ilusão de execução simultânea
- Um timer de hardware interrompe a execução de um processo em intervalos predeterminados,
  - cedendo o processador para outro processo na fila
- Nenhum processo monopoliza o processador por muito tempo,
  - evitando travamentos e atrasos perceptíveis para o usuário

Scheduler (Escalonador) e Prioridades

- responsável por gerenciar a alocação das fatias de tempo entre os processos
  - decide qual processo será executado a seguir
- pode atribuir prioridades diferentes aos processos
  - Processos com maior prioridade recebem mais tempo de CPU ou fatias de tempo maiores
    - garantindo que tarefas críticas sejam executadas mais rapidamente

- Algoritmos de Escalonamento: O scheduler implementa algoritmos de escalonamento para determinar a ordem de execução dos processos, levando em consideração fatores como prioridade, tempo de espera, tempo de execução estimado, etc. Exemplos: Round Robin (RR), Prioridade Estática, Prioridade Dinâmica, etc.

- Context Switching (Troca de Contexto): Quando o tempo de um processo expira, o scheduler realiza uma troca de contexto, salvando o estado atual do processo e carregando o estado do próximo processo a ser executado. Essa troca tem um custo, e fatias de tempo muito curtas podem aumentar a sobrecarga devido a trocas frequentes.

Sistemas Interativos de Tempo Compartilhado

- combinam multiprogramação com time-slicing para suportar múltiplos usuários simultaneamente
  - dando a cada usuário a impressão de ter um computador dedicado exclusivamente para si

- A combinação de multiprogramação e time-slicing permite que os usuários interajam com seus programas de forma responsiva
  - As respostas do sistema são rápidas o suficiente para dar a sensação de interação em tempo real

- Multitarefa: capacidade de um sistema executar múltiplos processos concorrentemente
  - dando a impressão de que estão sendo executadas simultaneamente

- Decomposição: a solução de um problema complexo é dividida em tarefas menores e independentes
  - que podem ser executadas em paralelo.
    - Necessário programar pensando nisso

- Concorrência: execução sobreposta ou intercalada dessas tarefas, gerenciada pelo SO

Correção de Programas Concorrentes

- Complexidade da Concorrência
  - A natureza não determinística da execução concorrente, devido às possíveis interações entre processos/threads,
    - torna complexa a escrita de programas corretos e livres de erros
  - A ordem exata de execução das instruções pode variar a cada execução,
    - dificultando a depuração e a reprodução de erros

Sincronização

- Mecanismos que controlam a ordem de execução de processos/threads e o acesso a recursos compartilhados,
  - evitando condições de corrida (race conditions) e
  - garantindo a consistência dos dados
- Exemplos: semáforos, mutexes, monitores, variáveis de condição

Comunicação

- Mecanismos que permitem a troca de informações entre processos/threads. Pode ser feita por:
  - Memória Compartilhada
    - Processos acessam uma área de memória comum para trocar dados. Requer mecanismos de sincronização para evitar conflitos de acesso
  - Troca de Mensagens
    - Processos trocam mensagens explicitamente através de canais de comunicação

Desafios da Correção

- Condições de Corrida (Race Conditions)
  - O resultado da execução depende da ordem em que os processos acessam recursos compartilhados,
    - levando a comportamentos inesperados
- Deadlocks (Impasses)
  - Dois ou mais processos ficam bloqueados indefinidamente,
    - esperando por recursos que estão sendo mantidos por outros processos no ciclo
- Inanição (Starvation)
  - Um processo é continuamente impedido de acessar um recurso, mesmo que ele esteja disponível,
    - devido a outros processos com maior prioridade ou a um escalonamento injusto

Técnicas de Depuração e Teste

- A depuração de programas concorrentes exige técnicas específicas, como
  - Análise de traços de execução
  - Testes de estresse
  - Uso de ferramentas de análise estática e dinâmica