Insper

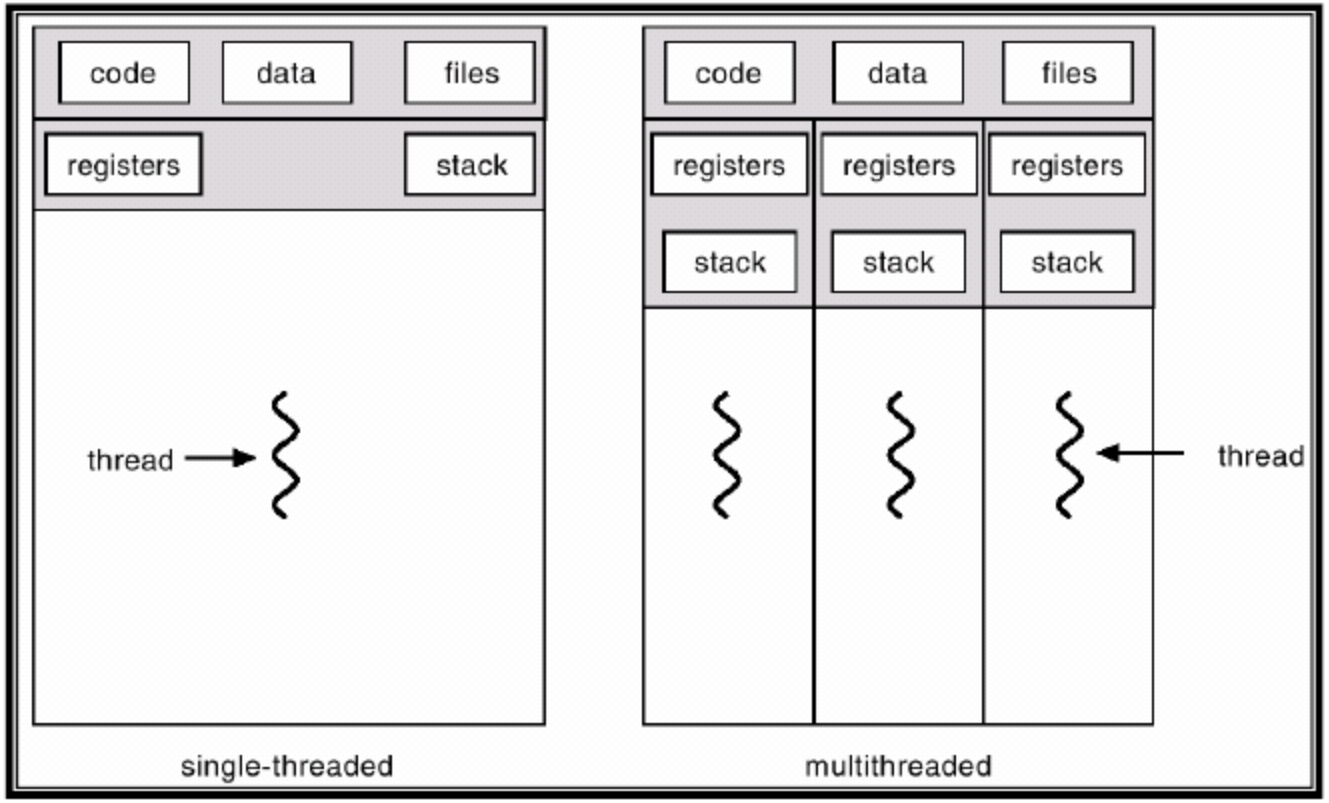
Sistemas Hardware-Software

Aula 25 – Programação concorrente II

2019 – Engenharia

Igor Montagner, Fábio Ayres <igorsm1@insper.edu.br>

Processos e threads



Insper

Processos e threads

- Processos
  - Comunicação entre processsos
  - **Possível distribuir em várias máquinas**

- Threads
  - Mais barato de criar e destruir
  - Sempre pertencem a um único processo
  - **Sincronização para acessar recursos compartilhados**

Troca de contexto ocorre de maneira igual nos dois casos!

POSIX threads

O padrão POSIX define também uma API de threads *(pthreads)* que inclui

- Criação de threads
- Sincronização (usando semáforos)
- Controle a acesso de dados (usando mutex)

Atividade (aula passada)

Vamos criar algumas threads e resolver problemas simples.

Parte 4

- Divisão de trabalho em várias threads
- Quatro argumentos
  - `double *vector`
  - `int start`
  - `int end`
  - `double sum`

- Somo as parciais após os `pthread_join`

Problemas limitados por CPU

- Roda tão rápido quanto a CPU puder
- Otimização de cache vale muito
- Faz pouca entrada/saída
  - Interage pouco com o sistema
- Pode ou não ter partes paralelas

Problemas limitados por CPU

- Dividimos um problemas em partes
- Cada parte é independente (em sua maioria)
- Juntamos os resultados no fim
- Pouca ou nenhuma sincronização

Tarefas paralelas (CPU-bound)

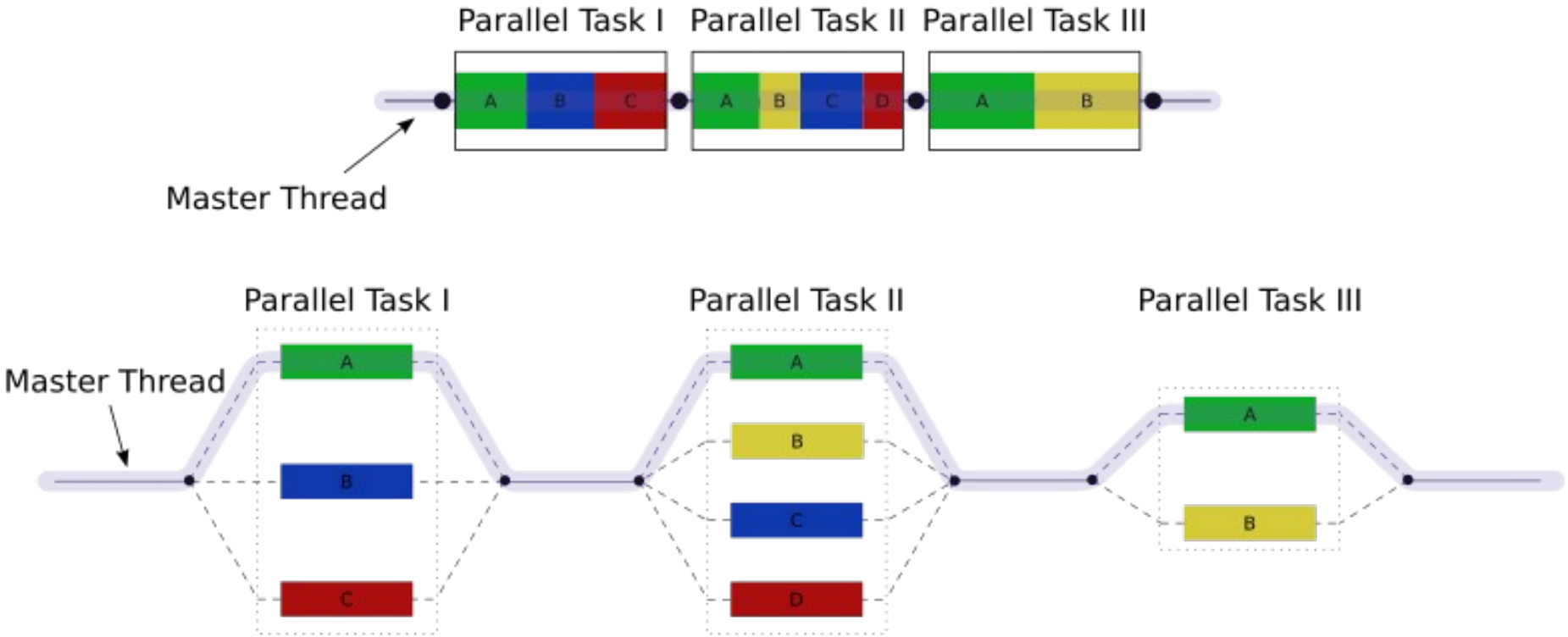


Figura: https://en.wikipedia.org/wiki/File:Fork_join.svg

Insper

Compartilhando dados

- E se precisarmos compartilhar dados?
  - Tarefas preenchendo um vetor
  - Leitura/escrita em variável
  - Tarefas heterogêneas
    - Fazem coisas diferentes
    - Mas usam mesmos dados

Atividade prática

- Exercício da parte 1 do handout

Race condition

saída do programa depende da ordem de execução das threads

- Acessos concorrentes a um recurso, com pelo menos uma escrita
- Nossa atividade tem esse problema!

Região crítica

parte(s) do programa que só podem ser rodadas por uma thread por vez

- Nenhum paralelismo é permitido em regiões críticas
- Evita acessos concorrentes, mas é gargalo de desempenho

Mutex (Mutual Exclusion)

Primitiva de sincronização para criação de regiões de exclusão mútua

- Lock – se estiver destravado, trava e continua
  - se não espera até alguém destravar
- Unlock – se tiver a trava, destrava
  - se não tiver retorna erro

Atividade prática

- Vamos usar mutexes para proteger o acesso a variável de soma.

- Contextualização: paralelização de buscas em arquivos

Insper

www.insper.edu.br