Computação Paralela
LCC/LERSI

Fundamentos de Computação Paralela

Computação Paralela

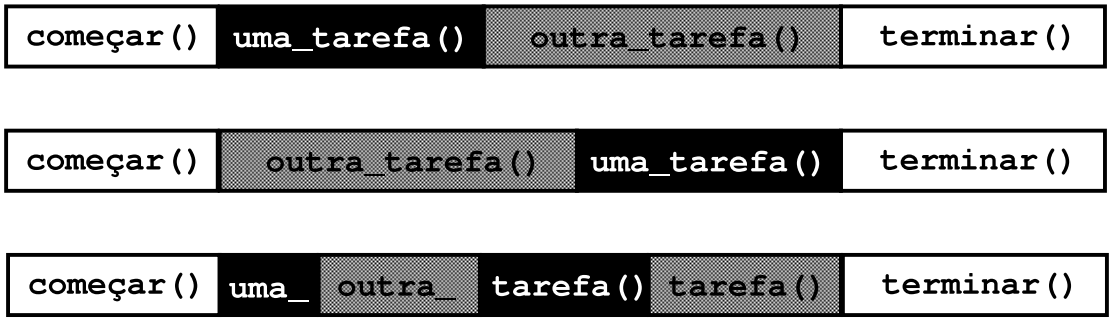
"Se um único computador consegue resolver um problema em 10 segundos, podem 10 computadores resolver o mesmo problema em 1 segundo?"

No que consiste a **Computação Paralela**?

- **Concorrência:** identificar partes da computação que possam ser executadas em qualquer ordem sem alterar o resultado final.
- **Scheduling:** distribuir de forma eficiente a computação pelos vários processadores disponíveis.
- **Comunicação e Sincronização:** desenhar a computação para ser executada em simultâneo nos vários processadores.

Concorrência ou Paralelismo Potencial

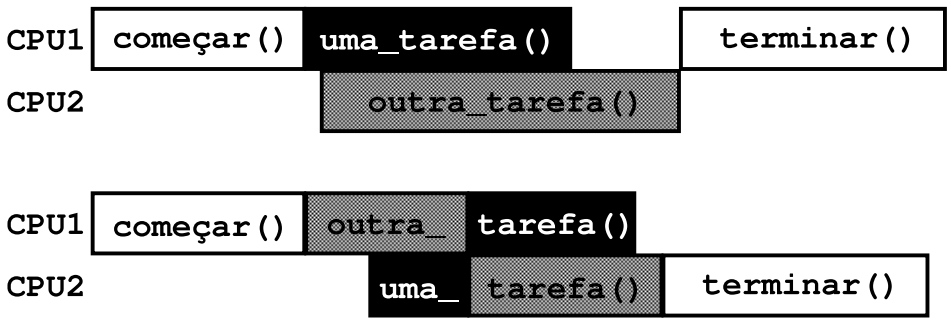
- Concorrência ou paralelismo potencial diz-se quando um programa possui **tarefas** (partes contíguas do programa) que podem ser executadas em qualquer ordem sem alterar o resultado final.



- Uma razão óbvia para explorar concorrência é conseguir reduzir o tempo de execução dos programas.

Paralelismo

- Paralelismo diz-se quando as tarefas de um programa são executadas em simultâneo numa máquina com mais do que um processador.



Paralelismo Explícito

O paralelismo diz-se explícito quando cabe ao **programador**:

- Anotar as tarefas para execução em paralelo.
- Atribuir (possivelmente) as tarefas aos processadores.
- Controlar a execução, indicando nomeadamente os pontos de sincronização.
- Conhecer a arquitectura da máquina de forma a conseguir melhor performance (aumentar localidade, diminuir comunicação, etc.)

Vantagens e inconvenientes:

- **(+)** Programadores experientes produzem soluções muito eficientes para problemas específicos.
- **(–)** O programador é o responsável por todos os detalhes da execução (*debugging* pode ser deveras penoso).
- **(–)** Pouco portável entre diferentes arquitecturas.

Paralelismo Implícito

O paralelismo diz-se implícito quando cabe ao **compilador** e ao próprio **sistema de execução**:

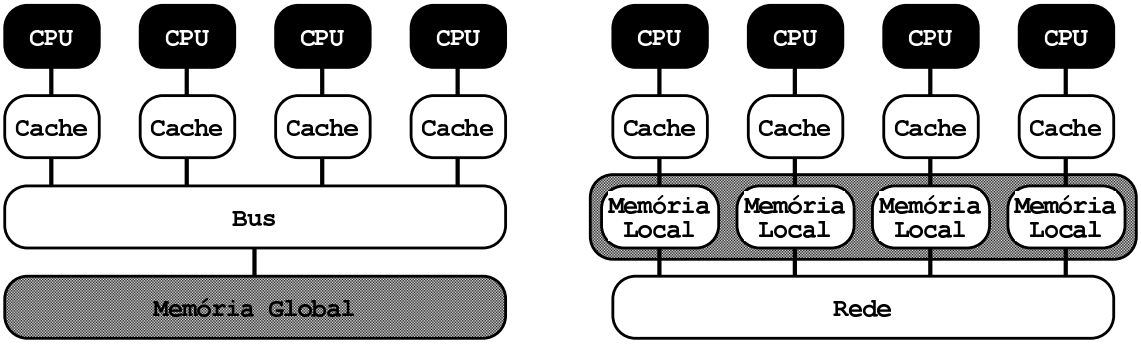
- Detectar o paralelismo potencial do programa.
- Atribuir as tarefas para execução em paralelo.
- Controlar e sincronizar toda a execução.

Vantagens e inconvenientes:

- **(+)** Liberta o programador dos detalhes da execução paralela.
- **(+)** Solução mais geral e mais flexível.
- **(–)** Difícil conseguir-se uma solução eficiente para todos os casos.

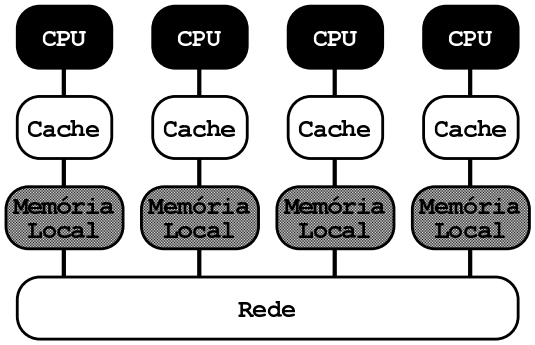
Computações Paralelas

- Uma computação diz-se paralela quando um programa é executado sobre uma máquina multiprocessador em que todos os processadores partilham o acesso à memória disponível (o mesmo endereço em processadores diferentes corresponde à mesma posição de memória).
- Existem duas grandes classes de máquinas multiprocessador:
  - **Centralized Multiprocessor** (ou *Uniform Memory Access Multiprocessor* (UMA) ou *Symmetrical Multiprocessor* (SMP))
  - **Distributed Multiprocessor** (ou *Nonuniform Memory Access Multiprocessor* (NUMA))



Computações Distribuídas

- Uma computação diz-se distribuída quando um programa é executado sobre uma máquina constituída por vários computadores (**Multicomputer**) cada um com acesso exclusivo à sua memória local (o mesmo endereço em processadores diferentes corresponde a posições de memória diferentes).



Decompor a Computação

Uma forma de diminuir a complexidade dos problemas é decompor a computação em partições mais pequenas. Existem duas estratégias principais:

- **Decomposição do Domínio:** determinar uma partição apropriada para os dados associados ao programa, e depois conceber um processo de associar a computação com a partição. Cada tarefa executa as mesmas operações sobre diferentes elementos do conjunto de dados.
  - Multiplicação de matrizes ($A_{n\times m} \times B_{m\times p} = C_{n\times p}$);
  - Algoritmo de ordenação *quicksort*.
- **Decomposição Funcional:** primeiro decompõe-se a computação a ser realizada e só depois se pensa nos dados. Cada tarefa executa diferentes operações sobre diferentes elementos do conjunto de dados.
  - Sucessão de fibonacci (fib(n) = fib(n - 1) + fib(n - 2));
  - Processamento de *querys*
    (Modelo="Civic" AND Ano="2001" AND (Cor="Azul" OR Cor="Branco")).

O Problema da Granularidade

"Como decompor a computação de modo a obter máxima performance?"

- O número e o tamanho das tarefas em que um problema é decomposto determina a **granularidade** da decomposição.
- A granularidade pode ser fina, média ou grossa.
  - Granularidade fina: decomposição num grande número de pequenas tarefas.
  - Granularidade grossa: decomposição num pequeno número de grandes tarefas.
- A granularidade da decomposição é importante porque:
  - O tempo de execução de uma tarefa tem de compensar os custos de criação, comunicação e sincronização (problema quando a granularidade é fina).
  - O tempo de ocupação dos processadores disponíveis deve ser o máximo possível (problema quando a granularidade é grossa).

O Problema da Granularidade

Multiplicação de matrizes: $A_{n \times m} \times B_{m \times p} = C_{n \times p}$

Os elementos de C são calculados por: $C_{i,j} = \sum_{k=1}^{m} A_{i,k} \times B_{k,j}$

Numa implementação, os elementos de C poderiam ser calculados por:

```
for (i = 1; i <= n; i++)
  for (j = 1; j <= p; j++)
    for (k = 1; k <= m; k++)
      C[i][j] += A[i][k] * B[k][j];
```

Granularidade da decomposição:

- Granularidade fina: cada processador calcula um elemento da matriz resultado (ciclo interior `k`).
- Granularidade média: cada processador calcula uma linha da matriz resultado (ciclos interiores `j` e `k`).

Padrões de Comunicação

A execução de uma tarefa pode envolver o acesso a dados calculados por outras tarefas. Para haver cooperação entre as tarefas concorrentes é necessário existir comunicação. Existem diferentes padrões de comunicação:

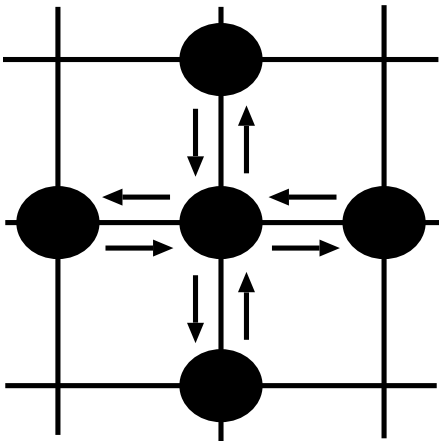
- **Comunicação Estruturada:** tarefas vizinhas constituem uma estrutura regular (e.g. árvore ou rede).
- **Comunicação Não Estruturada:** comunicação entre tarefas pode constituir um grafo arbitrário.

- **Comunicação Estática:** os parceiros de comunicação não variam durante a execução.
- **Comunicação Dinâmica:** a comunicação é determinada pela execução e pode ser muito variável.

Padrões de Comunicação

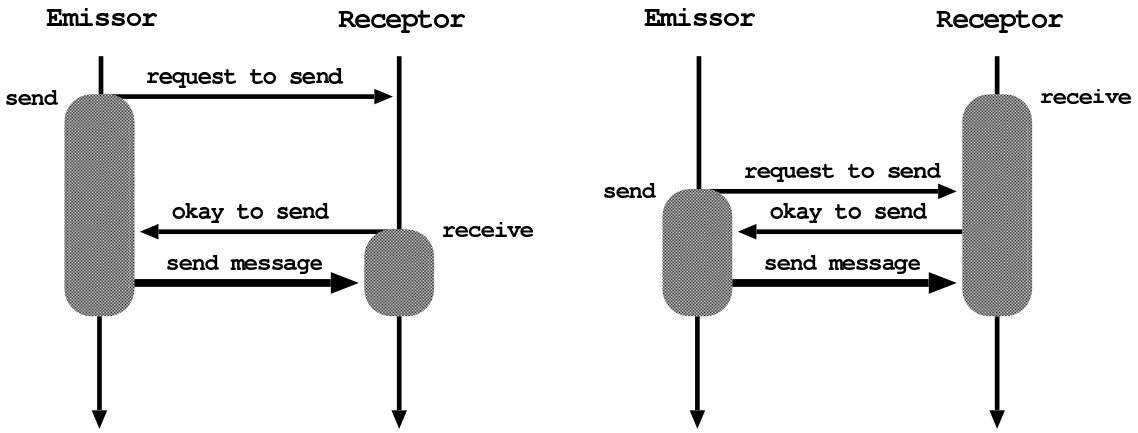
- **Comunicação Global:** todas as tarefas podem comunicar entre si.
- **Comunicação Local:** comunicação restrita a tarefas vizinhas.
  - Método de Jacobi de diferenças finitas.

$$X_{ij}^{t+1} = \frac{4X_{ij}^{t} + X_{i-1j}^{t} + X_{i+1j}^{t} + X_{ij-1}^{t} + X_{ij+1}^{t}}{8}$$



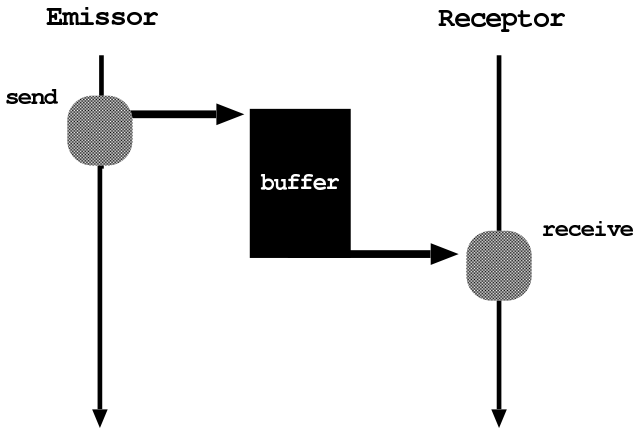
Padrões de Comunicação

- **Comunicação Síncrona:** as tarefas executam de forma coordenada e sincronizam na transferência de dados.
  - Protocolo das 3-fases ou *rendez-vous*: a comunicação apenas se concretiza quando as duas tarefas estão sincronizadas.



Padrões de Comunicação

- **Comunicação Assíncrona:** as tarefas executam de forma independente não necessitando de sincronizar para transferir dados.
  - *Buffering* de mensagens: o envio de mensagens não interfere com a execução do emissor.



Factores Limitativos de Performance

- **Código sequencial excessivo:** existem partes do código que são inerentemente sequenciais (iniciar/instanciar novas tarefas).
- **Granularidade da decomposição:** o número e tamanho das tarefas é importante porque o tempo que demoram a ser executadas tem de compensar os custos da execução em paralelo.
- **Carga não balanceada:** ter os processadores maioritariamente ocupados durante toda a execução é decisivo para o resultado global do sistema.
- **Atraso na sincronização:** a partilha de dados entre os vários processadores pode levar a problemas de contenção no acesso à memória.
- **Atraso na comunicação:** em sistemas de memória distribuída a comunicação é por natureza mais lenta.

Principais Modelos de Programação Paralela

Memória Partilhada

- Programação usando processos ou threads.
- Estratégias de decomposição: do domínio ou funcional.
- Granularidade da decomposição: fina, média ou grossa.
- Comunicação através de memória partilhada.
- Sincronização através de mecanismos de exclusão mútua.

Memória Distribuída

- Programação usando mensagens.
- Paralelismo mais frequente: decomposição do domínio com granularidade grossa.
- Comunicação e sincronização por troca de mensagens.